Question(s):	23		**Meeting, date:**	Geneva, 3-13 April 2006	
Study Group:	16	**Working Party:** 3	**Intended type of document** (R-C-D-TD):		D
Source:	Germany				
Title:	ITU-T JPEG-Plus Proposal for Extending ITU-T T.81 for Advanced Image Coding				

Contact:	Istvan Sebestyen Siemens Germany	Tel:+49-89-722-47230 Fax:+49-89-722-47713 Email:istvan.sebestyen@siemens.com
Contact:	Guido Vollbeding Independent JPEG Group Germany	Tel:+49-345-6851663 Fax: same (can be configured) Email:guido@jpegclub.org

Please don't change the structure of this table, just insert the necessary information.

Summary

This Proposal specifies three format extensions for digital compression and coding of still images according to ITU-T Rec. T.81 | ISO/IEC 10918-1 (JPEG-1) in order to solve some deficiencies of the original specification and thereby bringing DCT based JPEG back to the forefront of state-of-the-art image coding technologies.

The three extensions to be introduced are (1) an alternative coefficient scan sequence for DCT coefficient serialization, (2) a SmartScale extension in the Start-Of-Scan (SOS) marker segment for the sequential DCT mode, and (3) a Frame Offset definition in or in addition to the Start-Of-Frame (SOF) marker segment.

The introduction of the proposed specifications enables a new feature set which addresses five major requirements in application of Advanced Image Coding technologies today: (1) enhanced performance for image scalability, (2) provision for an efficient image-pyramid/hierarchical coding mode, (3) improved performance for competitive low-bitrate compression, (4) a seamlessly integrated lossless coding mode, and (5) performing basic lossless operations in compressed image data domain.

Keywords

Still-image coding, still-image compression, still images, image scalability, progressive coding, hierarchical coding, image-pyramid coding, low-bitrate compression, lossless compression.

Intellectual Property Rights

All specifications and algorithms presented in this Proposal are based on genuine perceptions by the author of this document which were not known before. The author claims NO Intellectual Property Right to these inventions, they are made available for free and unrestricted use in the public domain. The author is willing to transfer without charge any Intellectual Property Rights which may be associated with the presented inventions to the committee which approves this specification.

CONTENTS

Page

ITU-T JPEG-Plus Proposal for Extending ITU-T T.81 for Advanced Image Coding

1 Scope

This Proposal is applicable to continuous-tone, greyscale or colour, digital still-image data.
It enhances T.81 technologies by providing Advanced Image Coding features.

This Proposal

- specifies alternative coefficient scan sequences for DCT coefficient coding;
- defines a SmartScale extension in the Start-Of-Scan (SOS) marker segment;
- specifies a Frame Offset definition in or in addition to the SOF marker segment.

The provisions of ITU-T Rec. T.81 | ISO/IEC 10918-1 shall apply to this Proposal with the exceptions, additions, and deletions given in this Proposal.

2 Introduction

JPEG-Plus is the designed name from the author of this Proposal for a future JPEG update for Advanced Image Coding features.

The name summarizes what one would expect from a proper JPEG update: a superset framework which includes the old modes (T.81/JPEG-1) as a subset for backwards compatibility, similar as known with the computer programming languages C and C++. So one could also think of JPEG+ or JPEG++, but since JPEG is not a programming language (well, not really), the author thinks that JPEG-Plus is the best name.

Filename extensions for files which carry the new data streams could be .jpp for example.

As long as the new format can't be approved by the JPEG committee (as "Joint" stands for ISO and ITU), but only by ITU, for example, alternatives could be used such as .ipg (for ITU Photographic Experts Groups, or International Photographic Experts Group, or Independent Photographic Experts Group).

The new features presented in this Proposal provide noticeable advantages to a wide range of image coding applications where JPEG-1 (ITU-T Rec. T.81) was successfully used so far and beyond, while the additional specification and implementation effort is minimal.
Thus the formalized standardization of the given Proposal by a standardization committee like ITU, and the provision of a widely usable free reference implementation in collaboration with the Independent JPEG Group, which was a key to the success of the JPEG-1 standard, would enable new marketing and business activities for the benefit of a wide range of participants.

Backwards compatibility to the existing JPEG format can easily be retained by implementations of the extended JPEG-Plus format, in the sense that extended decoders or encoders can easily read or optionally output old JPEG files, respectively, and via lossless transcoding it is also possible to convert old JPEG files to new capabilities or vice versa!

3 Overview

Regarding the description of the proposed specifications and corresponding features, this document is organized in the form of a Top-Down approach.
This means that we start with describing the final specifications and features, while giving more detailed explanations of underlying properties and algorithms later.

The three proposed specifications are introduced in the following three chapters (4-6) with description of their corresponding features. The three specifications are:

1) An alternative coefficient scan sequences for DCT coefficient coding.

2) A SmartScale extension in the Start-Of-Scan (SOS) marker segment.

3) A Frame Offset definition in or in addition to the SOF marker segment.

The first two specifications enable the following four Advanced Image Coding features:

1) enhanced performance for image scalability;

2) efficient image-pyramid/hierarchical coding;

3) improved low-bitrate compression;

4) seamlessly integrated lossless coding.

The third specification enables the following additional Advanced Image Coding feature:

5) unrestricted lossless cropping and transformation operations in the compressed domain.

The SmartScale extension also enables a lossless (without quality degrading recompression) rescale feature which will be described in the corresponding chapter (5).

The first two specifications and corresponding features are derived from the new DCT scaling algorithms and features as currently being introduced for use with existing JPEG into the next official Independent JPEG software release (v7 in this year). See also http://jpegclub.org for more information and preliminary results.

These new DCT scaling algorithms and features are described in Annex A.

Annex B contains a short description of the underlying "fundamental DCT property for image representation". This property was found by the author during implementation of the new DCT scaling features and is after his belief the most important discovery in digital image coding since releasing the JPEG standard in 1992.

The third specification is derived from implementation and application of lossless transformation (90 degree rotation etc.) and cropping features in the IJG jpegtran utility for lossless transcoding of JPEG files.

4 Alternative coefficient scan sequences for DCT coefficient coding

4.1 Enhanced performance for image scalability

Scalability is a key feature in image processing (see also Annex B).
The new IJG v7 DCT scaling features work well (see also Annex A), but not optimal due to constraints in the DCT coefficient serialization.
The current JPEG standard has provision only for the diagonal zig-zag sequence.
For optimal utilization of DCT scaling, an alternative, sub-block-wise, scan sequence as follows is more appropriate, since lower resolutions can be derived directly from coefficient sub-blocks:

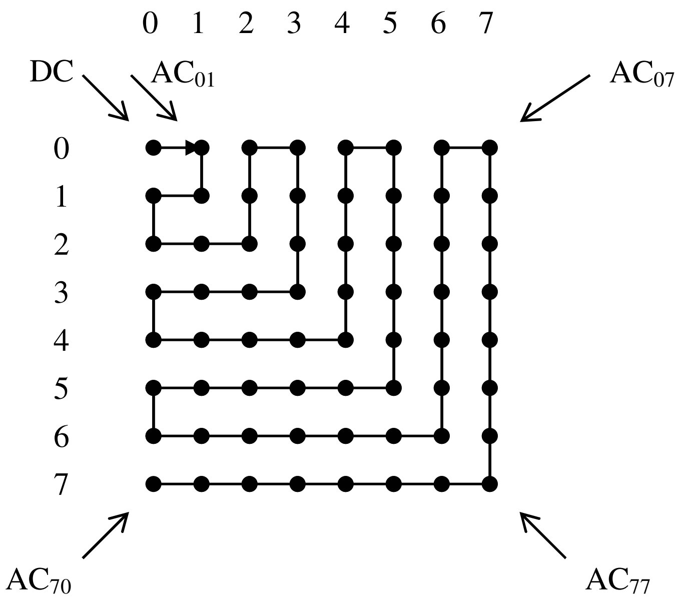


Figure 4-1 – Alternative sub-block-wise coefficient scan sequence

Compare Figure 4-1 with Figure 5 in T.81 | ISO/IEC 10918-1 (diagonal zig-zag scan).
An alternative scan sequence is very easy to implement in the IJG library, since the access to coefficients is handled via a table-lookup.
Thus, no changes in the core coding functions are necessary, only another index table must be provided.

Alternative scan sequences can be provided in the JPEG(-Plus) file by specification in an optional JPEG marker segment in the file header.
Either the selection of predefined tables is possible, or the specification of arbitrary user-defined tables similar to the quantization tables.
Section 4.3 proposes a concrete specification format for alternative coefficient scan sequences.

Alternative scan sequences for DCT coefficient coding were also introduced in the MPEG-2 video coding standard, in order to adapt to interlaced video modes in this case.

According to hints in the Pennebaker and Mitchell JPEG book, the diagonal zig-zag sequence in the current JPEG standard was chosen rather arbitrarily, and different schemes should not have significant impact on coding efficiency, especially in the arithmetic coding case according to the authors.

Since the scalability properties of the DCT (see Annex A) were not known by the authors of the JPEG standard, they did not make provision for an appropriate scan selection, so this feature must be added in a standards update.

Here is an example which depicts the advantage of the alternative sequence over the current diagonal scan when half-size downscaling the image:

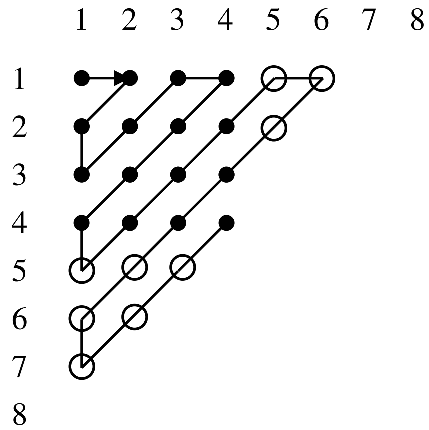


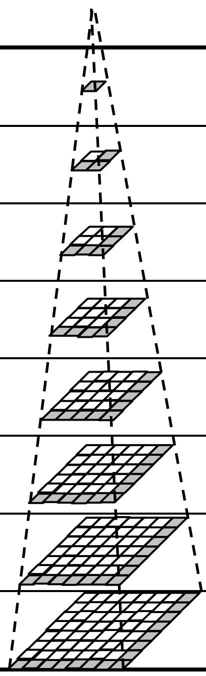
Figure 4-2 – Coefficient scan sequence for half-size downscale with diagonal scan

The figure shows the sequence of coefficients to scan for half-size downscaling with the given diagonal scan.
For the half-size downscaled image, only the upper left 4x4 block of coefficients is required (symbol "●"). Due to the diagonal scan, we must also scan or skip some unnecessary coefficients (symbol "o") in the sequence. The current DCT scaling implementation is already optimized in so far that it runs only to the required edge coefficient in the block (4,4) and skips the rest of 8x8 coefficients of the full block (left blank in the figure). But still, the unnecessary "o" coefficients remain.
With the above sub-block-wise alternative scan this problem is easily solved – all coefficients are arranged in such a way that no unnecessary coefficients occur in a sub-block sequence.

4.2 Efficient image-pyramid/hierarchical multi-resolution coding

The alternative scan sequence given in Figure 4-1 not only optimizes the scaling performance, but it also enables another important capability:
With the given standard Progressive JPEG mode (Spectral Selection feature) and the new alternative coefficient scan sequence we can construct perfect "image pyramids" which makes the cumbersome, inefficient, and therefore rarely implemented Hierarchical mode in the given JPEG standard obsolete.
We can build progressive scan sequences (based on the Spectral Selection feature) with successive resolution enhancement. Usually the Progressive JPEG mode allowed only successive quality enhancement at a given resolution, and that's why the Hierarchical mode was introduced in the JPEG standard.
No other changes to the specification or implementation are required to enable this new capability.
This capability can be integrated in a corresponding framework for variable resolution (image pyramid/hierarchical) handling.
The following table shows the progressive scan parameters for a full multi-resolution progression:

Table 1 – Progressive scan parameters for full multi-resolution progression



	Scan Nr.	Ss	Se	Resolution Scale Factor
	1	0	0	1/8
	2	1	3	2/8 = 1/4
	3	4	8	3/8
	4	9	15	4/8 = 1/2
	5	16	24	5/8
	6	25	35	6/8 = 3/4
	7	36	48	7/8
	8	49	63	8/8 = 1

The parameters can be derived from the following table which specifies the alternative sub-block-wise coefficient scan index table according to Figure 4-1:

Table 2 – Alternative sub-block-wise coefficient scan index table

Scan Nr.	1	2	3	4	5	6	7	8
1	0	1	8	9	24	25	48	49
2	3	2	7	10	23	26	47	50
3	4	5	6	11	22	27	46	51
4	15	14	13	12	21	28	45	52
5	16	17	18	19	20	29	44	53
6	35	34	33	32	31	30	43	54
7	36	37	38	39	40	41	42	55
8	63	62	61	60	59	58	57	56

It is of course not necessary to use the full multi-resolution progression. Several scans can be combined and thereby some resolutions skipped if appropriate in application. The Progressive mode (particularly the Spectral Selection mode) parametrization of original T.81 provides sufficient flexibility here for various application preferences.

4.3 Specification of alternative coefficient scan sequences

We propose here a particular specification for the selection of alternative coefficient scan sequences by extending the given DQT (Define Quantization Table) marker segment in a backwards compatible way.
The advantage of this specification is that no extra marker has to be introduced, that implementations are easy to adapt for this new selection (especially in an evaluation phase), and that different components may use different coefficient scan sequences.
The coefficient scan sequences shall be associated with the corresponding quantization table identifiers (slots).

The DQT marker syntax is specified in Section B.2.4.1 "Quantization table-specification syntax" in T.81 as follows (Figure B.6 and Table B.4):

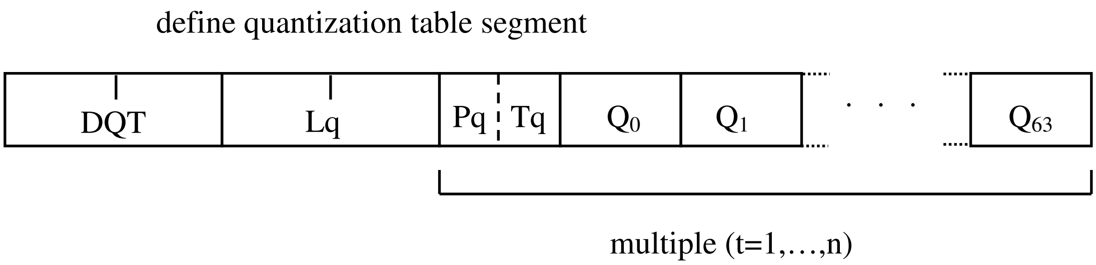


Figure 4-3 – Quantization table syntax per T.81

DQT : define quantization table marker = 0xFFDB.
Lq : quantization table definition length (variable, see table below).
Pq : quantization table element precision (0 = 8-bit Q_k; 1 = 16-bit Q_k).
Tq : quantization table identifier.
Q_k : quantization table element sequence in diagonal zig-zag-order.

Table 3 – Quantization table-specification parameter sizes and values per T.81

parameter	size (bits)	Values			
		sequential DCT		progressive DCT	lossless
		baseline	extended		
Lq	16	$2+\sum_{t=1}^{n}(65+64\times Pq(t))$			undefined
Pq	4	0	0, 1	0, 1	undefined
Tq	4	0-3			undefined
Q_k	8, 16	1-255, 1-65535			undefined

The parameter Pq is a local flag value where only one of four available bits is used (the least significant bit 0, mask 1). We can use the three other bits for extension purposes.

We define bit 1 (mask 2) as follows:

Pq bit 1 = 0 : use diagonal zig-zag sequence per T.81 Figure 5 and Figure A.6.
else : use alternative sub-block-wise sequence per Figure 4-1 and Table 2.

Furthermore we define bit 2 (mask 4) as follows:

Pq bit 2 = 0 : no change.
else : insert 64 bytes before Q_k values for custom coefficient scan sequence definition.

The new extended specification looks as follows ("&" is the bitmask operator):

Table 4 – Extended quantization table-specification parameter sizes and values

<table>
<tr><th rowspan="3">parameter</th><th rowspan="3">size (bits)</th><th colspan="4">Values</th></tr>
<tr><th colspan="2">sequential DCT</th><th rowspan="2">progressive DCT</th><th rowspan="2">lossless</th></tr>
<tr><th>baseline</th><th>extended</th></tr>
<tr><td>Lq</td><td>16</td><td colspan="3">$2+\sum_{t=1}^{n}(65+64\times(Pq(t)\,\&\,1)+16\times(Pq(t)\,\&\,4))$</td><td>undefined</td></tr>
<tr><td>Pq</td><td>4</td><td>0</td><td>0,1, 2,3, 4,5</td><td>0,1, 2,3, 4,5</td><td>undefined</td></tr>
<tr><td>Tq</td><td>4</td><td colspan="3">0-3</td><td>undefined</td></tr>
<tr><td>S_k</td><td>0, 8 *64</td><td colspan="3">0-63</td><td>undefined</td></tr>
<tr><td>Q_k</td><td>8, 16 *64</td><td colspan="3">1-255, 1-65535</td><td>undefined</td></tr>
</table>

This specification provides the selection of the default sub-block-wise coefficient scan sequence per Figure 4-1 and Table 2 without any expansion in the size of the data stream compared to the old diagonal zig-zag scan. Custom (downloadable) coefficient scan sequences may be defined optionally to allow adaption for specific purposes and applications (see Section 4.5 for an example).

Table 4 shall replace Table B.4 from T.81 | ISO/IEC 10918-1.

JPEG files with diagonal scan can be losslessly transcoded to an alternative scan and vice versa.

4.4 Efficient low-bitrate compression

Two other image coding features can also be accomplished with the new DCT scaling options:

- "Low Bit-rate Compression":
  Use downsampled encoding and upsampled decoding for better performance in low bit-rate domain.
  (Uses higher-order DCT transforms for higher correlation.)

- "Lossless Compression":
  Use upsampled encoding and downsampled decoding to accomplish a lossless compression scheme.
  (Uses lower-order DCT transforms to avoid computing loss.)

The second feature will be described in the next section.

Strictly speaking, the low bit-rate compression mode is not provided with the given specification extension for alternative coefficient scan sequences. It is already provided with the new IJG v7 direct DCT scaling algorithms and features (see Annex A). For more convenient utilization of the low bit-rate coding mode we will introduce in Chapter 5 the SmartScale extension feature.

The new IJG v7 library provides features for direct rescaling of images while compressing to and decompressing from JPEG. For this purpose, different size (NxN, with N=1…16) FDCT and IDCT algorithms are utilized in order to produce different spatial size output from the usual 8x8 DCT coefficient block in the JPEG file, or to produce 8x8 DCT coefficient blocks from different size spatial pixel blocks, respectively.

The higher-order DCTs (N>8, up to N=16 for a factor of 2 rescaling) are used instead of the usual 8x8 DCT for upscaling via decoding and downscaling via encoding. The algorithms are optimized and very efficient, and since the internal DCT coefficient block size remains the standard 8x8 JPEG size, the adaption in the implementation does not require change of the basic data block structures for DCT coefficients, and the features and formats are fully compatible with the standard 8x8 DCT based JPEG system.

The use of higher-order (up to 16) DCT algorithms makes the features especially suited for low bit-rate domain compression application, since better data correlation can be exploited by using higher block size in the spatial pixel domain (up to 16x16). When encoding, the higher order DCT coefficients beyond 8 are discarded, while only the lower order 8x8 coefficients are recorded. This corresponds to a factor of 8/N (1/2 for N=16) downscale when the file would be decompressed normally. However, with the complementing decoder rescale option we can decode the file back to the original resolution by using the corresponding upscale option of N/8 (2/1 in case of N=16). The same size inverse DCT is used in this case to produce the original image resolution, while higher order coefficients beyond 8 are set to zero in the inverse transformations.

So the low-bitrate compression mode consists of following steps:

(1) choose N = 9…16;
(2) cjpeg (compress jpeg) –scale 8/N (downscale);
(3) djpeg (decompress jpeg) –scale N/8 (upscale).

We will introduce in Chapter 5 a SmartScale extension to make application of this mode more convenient.

The same basic idea was already recognized and utilized before, although in a less efficient way, in the National Imagery Transmission Format Standard (NITFS) by the National Imagery and Mapping Agency (NIMA), as specified in the following document:

> http://www.ismc.nima.mil/ntb/baseline/docs/n010697/bwcguide25aug98.pdf

It introduces in Chapter 5 a method called "Downsample JPEG Compression (NIMA Method 4)":

> The specification for downsample JPEG is the standardized result of field trials of the approach also known as “NIMA Method 4.” The NIMA Method 4 approach provides a means to use existing lossy JPEG capabilities in the field to get increased compression for use with low bandwidth communications channels. This gives the field a very cost-effective approach for a critical capability during the period that the JPEG 2000 solution is being resolved.

NIMA Method 4 specifically correlates to a selection option (Q3) within downsample JPEG that provides a very useable tradeoff between file compression and the resulting loss in quality.

5.1.2.1 Image downsampling
The downsample JPEG algorithm encoder utilizes a downsampling procedure to extend the low bit-rate performance of the NITFS JPEG algorithm described in MIL-STD-188-198A. Figure 5-2 illustrates the concept. The downsampling preprocessor allows the JPEG encoder to operate at a higher bit-rate on a smaller version of the original image while maintaining an overall bit-rate that is low.

The creators of this standard did not know about the scaling properties of the DCT, or they utilized a closed “black-box” JPEG system, and so they used a pre- and post-processor to achieve the same effect as our new direct DCT scaling features achieve in a single step!
See also "APPENDIX F - ENGINEERING DESIGN DETAILS FOR THE DOWNSAMPLE JPEG SYSTEM":

F.2 DOWNSAMPLE JPEG SYSTEM MODEL
The Downsample JPEG compression algorithm achieves very low bit rate compression using the scheme shown in Figure F-1. Decimation of the original image is used to achieve bit rates beyond what JPEG can accomplish alone (0.5-0.8 bits/pixel) due to the fixed 8x8 block size encoding structure. In this algorithm, the adverse effects of downsampling (e.g., aliasing and blurring) are traded-off with JPEG artifacts (e.g., blocking) by adjusting the relative compression contributions from each module. The quality of the reconstructed image after JPEG decompression and upsampling has been demonstrated to be competitive with several "state-of-the-art" low bit rate compression algorithms.

This shows that the creators of this Military JPEG specification were bright and intelligent.
They did not know about the fundamental DCT property and thus could not find the optimal solution, but they were already on the right track.

4.5 Seamlessly integrated lossless coding mode

The downsampled encoding and upsampled decoding with the new DCT scaling options provide the low bit-rate compression mode based on the use of higher-order ($8<N\leq16$) FDCT and IDCT algorithms as described in the previous section.

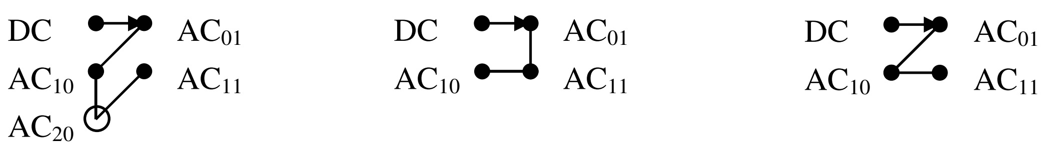
In the other direction, selecting upsampled encoding and downsampled decoding uses lower-order ($1\leq N<8$) FDCT and IDCT algorithms and leads us to the option of supporting a seamlessly integrated lossless coding mode.

Especially interesting in this regard are the high cjpeg scalings of 8/1 and 4/1:
Since the used 1x1 and 2x2 FDCT/IDCT algorithms (see Annex A Table A-1) do not involve any transcendent or fractional multiplication (with appropriate quantization), these modes can be used to support a true lossless coding scheme within the standard 8x8 DCT JPEG system.

In particular, cjpeg -scale 8/1 just sets each source sample as DC value (properly scaled) in the 8x8 DCT block and all ACs zero. While not very interesting for scaling purposes, we can decode that image with djpeg -scale 1/8 (which just derives each output sample from the DC value) and get an

exact copy of the source image (with appropriate quantization factor, which should be 8 for the DC in this case, and avoiding YCbCr/RGB conversion).
This also works under similar conditions (q-factor=2 for the DC and 3 ACs) with the cjpeg –scale 4/1 and djpeg -scale 1/4 case, because the 2x2 FDCT/IDCT algorithms are just simple sum/diff schemes without transcendent or fractional multiplication.

So while these features work in principle already with the given standard conforming DCT scaling implementation, we still have to investigate how the entropy coding can be optimized for efficient lossless image coding.
With the given implementation we would have to code large runs of zeros for the ACs which is not optimal, particularly in the sequential JPEG mode.
We can work around this partially by using the progressive JPEG mode to split DC and AC scans and omitting obsolete zero run AC scans.
For example, in the 8/1 scale case we simply have to code the DC scan and can omit any AC scans altogether. The DC-only coding corresponds to a basic DPCM lossless coding scheme. No adaptions in the given implementation would be necessary for this approach, just select an appropriate progressive scan script.
However, the fact that this approach corresponds to a basic 1D-DPCM coding scheme suggests that this might not yet be the optimal solution for lossless image coding, since no provision is being made for pixel correlation in the vertical direction to obtain a true 2D image coding scheme.
The alternative 4/1 scale case seems more interesting in this regard, since 2D image correlation is thereby utilized to some extent (2x2 block transformation).
However, we cannot construct a suitable progressive scan sequence for this purpose with the given diagonal zig-zag scan. With the new alternative coefficient scan sequence we can construct a suitable scan, as depicted in the following figure:



Diagonal scan | Sub-block-wise scan | Custom alternative scan

Figure 4-4 – Various coefficient scan sequences for 2x2 sub-block size coding

With the diagonal scan we have an obsolete zero element AC_{20} in the sequence which impairs the coding performance.
With the new default sub-block-wise scan we can construct a suitable progressive scan script for optimal coding of the 2x2 coefficient sub-block size (Scans Nr. 1 and 2 in Table 1 and Table 2 in Section 4.2, while omitting the rest).
A custom alternative scan can be used to mix properties of the two predefined scans.

5 SmartScale extension in the Start-Of-Scan (SOS) marker segment

The previous two sections introduced two other image coding features as application of the new direct DCT scaling options.
An inconvenience for appropriate application of these features is that we have to remember somehow the scaling factor we used during encoding in order to apply the correct reverse scaling factor during decoding for obtaining the original image resolution.
Furthermore, the new lossless coding mode required that we use the progressive JPEG mode with incomplete progressive scan sequences for efficient entropy coding.

The SmartScale extension provides a simple mechanism to solve these inconveniences and thereby making the application of the new features more convenient for common usage.

The Start-Of-Scan (SOS) marker segment contains, among other fields, the two entries Ss and Se for specifying the start and end of spectral selection in the progressive JPEG mode.
For the sequential JPEG mode, the two entries shall be set to fixed values (Ss=0 and Se=63), and thus carry no further information for the decoding process.
This is the point for applying the SmartScale extension.
The SmartScale extension allows to assign other values than 63 to the Se parameter in the sequential DCT JPEG mode and specifies a corresponding adapted encoding and decoding process.

For describing the new specification we assume that we have selected the alternative sub-block-wise coefficient scan sequence per Figure 4-1 and Table 2.

Now assume that we are performing an upscaled encoding process, for example with scale factor 4/1 with 2x2 sub-block size for lossless coding application.
The sub-block 2x2 FDCT is performed in such a way that all coefficients outside the 2x2 sub-block are set to zero. The normal entropy encoding in the sequential DCT mode would thus produce large runs of zero until reaching the End-Of-Block (EOB) condition with the last coefficient index 63.
Instead of coding the obsolete large zero runs, it would be better to adapt the encoding in such a way that the AC coefficient index (1,1) (block number 2,2; Se=3) becomes the End-Of-Block position instead of 63, similar to the coding process in the progressive mode (except that we code only a single spectral selection scan here starting with and including the DC).
So we can set Se=3 in this case and adapt the entropy encoding and decoding processes with corresponding EOB handling similar to the spectral selection progressive mode.
In general, we would just use the corresponding value of Se as loop limit and EOB position in the entropy coding functions instead of the fixed value 63.

But there is another important adaption to make in the SmartScale case compared to the normal scaling case: The image height and width fields in the SOF (Start-Of-Frame) marker segment shall not be scaled, but instead shall remain the same as in the source image. Thus, the SmartScale mode shall not change the nominal spatial resolution of the source or destination image, respectively. The implementations must take care to handle the different dimensions appropriately, but otherwise no major changes in the internal data structures are necessary.
When decoding a SmartScale extended data stream (Se <> 63), the implementation must take care to select and perform the corresponding DCT size processing and derive appropriate scaled dimensions according to the Se sub-block position for internal data structures which can still be based on 8x8 DCT coefficient blocks. Only the output image dimensions shall match the corresponding SOF entries.

So far, the following Se values would be allowed with the alternative coefficient scan sequence per Figure 4-1 and Table 2 (see also progressive scan parameters in Table 1):

Table 5 – SmartScale parameter settings for Se<64

JPEG(-Plus) processing mode	**SmartScale compression factor**	**SmartScale decompression factor**	**DCT size (order)**	**Ss**	**Se**
sequential DCT	8/1	1/8	1 x 1	0	0
sequential DCT	4/1	1/4	2 x 2	0	3
sequential DCT	8/3	3/8	3 x 3	0	8
sequential DCT	2/1	1/2	4 x 4	0	15
sequential DCT	8/5	5/8	5 x 5	0	24
sequential DCT	4/3	3/4	6 x 6	0	35
sequential DCT	8/7	7/8	7 x 7	0	48
sequential DCT	1/1	1/1	8 x 8	0	63

The encoder accepts a SmartScale compression factor similar to the usual scale, derives the corresponding Se value, adapts the encoding process accordingly, and sets unscaled (original) spatial source image dimensions in the SOF marker.
The decoder finds a variable Se parameter in the SOS marker segment, derives the corresponding SmartScale factor, adapts the decoding process accordingly, and outputs the image in the same source resolution as specified in the SOF marker.
Different from the usual scaling, the decoder does not need to be given the reverse scaling option. The decoder examines the Se value in the SOS marker segment and can derive all necessary information to process the data and output the image in the original resolution.

Thus, for lossless coding application we need only give an appropriate option to the compressor (cjpeg) side, and the decompressor will automatically output the correct source resolution without further option:

(1) cjpeg –smartscale 8/1
(2) djpeg

or

(1) cjpeg –smartscale 4/1
(2) djpeg

Or even more convenient, we can just introduce a new –lossless n (n = 1 or 2) option to cjpeg which is assigned to –smartscale 8/n, and then just say:

(1) choose n = 1 or 2 (n=1 for 1D, n=2 for 2D processing mode)
(2) cjpeg –lossless n
(3) djpeg

For lossless operation, the –lossless option must adapt some other things in the compression process: Appropriate selection of quantization factors (see Section 4.5) and suppressing colorspace conversion (the IJG implementation supports the use of a direct RGB mode by using a special Adobe application marker which could be used for this purpose).

The SmartScale extension for other scaling factors in application for the low-bitrate coding mode is derived similarly as follows.
In this case we set the Se parameter with values > 63 to indicate the internal scaling and use of higher-order DCT algorithms.
To find appropriate values, we extend the alternative sub-block-wise coefficient scan sequence per Figure 4-1 and Table 2 from the 8x8 to a 16x16 block size:

Table 6 – Alternative sub-block-wise coefficient scan index table for size 16x16

Blk. Nr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	1	8	9	24	25	48	49	80	81	120	121	168	169	224	225
2	3	2	7	10	23	26	47	50	79	82	119	122	167	170	223	226
3	4	5	6	11	22	27	46	51	78	83	118	123	166	171	222	227
4	15	14	13	12	21	28	45	52	77	84	117	124	165	172	221	228
5	16	17	18	19	20	29	44	53	76	85	116	125	164	173	220	229
6	35	34	33	32	31	30	43	54	75	86	115	126	163	174	219	230
7	36	37	38	39	40	41	42	55	74	87	114	127	162	175	218	231
8	63	62	61	60	59	58	57	56	73	88	113	128	161	176	217	232
9	64	65	66	67	68	69	70	71	72	89	112	129	160	177	216	233
10	99	98	97	96	95	94	93	92	91	90	111	130	159	178	215	234
11	100	101	102	103	104	105	106	107	108	109	110	131	158	179	214	235
12	143	142	141	140	139	138	137	136	135	134	133	132	157	180	213	236
13	144	145	146	147	148	149	150	151	152	153	154	155	156	181	212	237
14	195	194	193	192	191	190	189	188	187	186	185	184	183	182	211	238
15	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	239
16	255	254	253	252	251	250	249	248	247	246	245	244	243	242	241	240

Now we can assign the following settings for Se > 63:

Table 6 – SmartScale parameter settings for Se>63

JPEG(-Plus) processing mode	SmartScale compression factor	SmartScale decompression factor	DCT size (order)	Ss	Se
sequential DCT	8/9	9/8	9 x 9	0	80
sequential DCT	4/5	5/4	10 x 10	0	99
sequential DCT	8/11	11/8	11 x 11	0	120
sequential DCT	2/3	3/2	12 x 12	0	143
sequential DCT	8/13	13/8	13 x 13	0	168
sequential DCT	4/7	7/4	14 x 14	0	195
sequential DCT	8/15	15/8	15 x 15	0	224
sequential DCT	1/2	2/1	16 x 16	0	255

Similar approach as described above for adapting the compression and decompression processes holds for these cases.
Ss and Se fields are reserved with 8 bits data size each in the SOS, so the new values fit perfectly with maximum value 255.
The entropy coding and EOB handling is like normal 8x8 block size with Se=63 independent from actual Se in these cases since recorded DCT coefficients are limited to the standard 8x8 JPEG block size. Se determines only the internal DCT size (order) and corresponding data scaling, according to the assignment given in the table.

SOF always contains the original source and destination spatial image dimensions, respectively, in the SmartScale mode.
This is an advantage over the approach described in Section 4.4 to the low-bitrate mode with the usual scaling option, since the usual downscaled compression could lose precision of image dimensions when first reduced by scale and then expanded by scale.
With the SmartScale extension, original spatial image dimensions are retained exactly in any case.

Now the application of the low-bitrate compression mode simplifies as follows:

(1) choose N = 9…16
(2) cjpeg –smartscale 8/N
(3) djpeg

For more convenience we can introduce a new –lowbitrate n option (n=1…8) to cjpeg which is assigned to –smartscale 8/(8+n). The parameter n serves as a lowbitrate compression level: n=1 is lowest lowbitrate compression level, n=8 is highest lowbitrate compression level.
Then we have simply and conveniently:

(1) choose n = 1…8 low-bitrate compression level (1=min, 8=max)
(2) cjpeg –lowbitrate n
(3) djpeg

5.1 Using Smartscale extension for lossless rescale option

It is often desired in image compression applications that a given compressed image should be reduced in resolution and corresponding data size without fully recompressing and thereby degrading the image.
So one wants another compressed image from a given compressed image with reduced resolution and correspondingly reduced data size.
Unfortunately, this application is not directly possible with the given DCT scaling features. The given DCT scaling features work only for transforming from compressed DCT to spatial domain or vice versa. Therefore, the rescaling options are only available in cjpeg and djpeg applications, but not in jpegtran application for lossless transcoding.
With the SmartScale extension, however, we can provide the desired feature also for the jpegtran application. That is, we can introduce in jpegtran a corresponding option "-smartscale N/8" and change losslessly (without full recompression) the nominal spatial resolution of a given compressed image.
For N<8 (downscale) we would simply retain the NxN sub-block from the full 8x8 DCT coefficient block and re-encode the result with corresponding Se/EOB setting with the entropy coding process only. In this case we have to adapt the image dimension fields in the SOF marker accordingly, so that the nominal spatial resolution of the image changes appropriately. The compressed data size reduces accordingly by omitting the higher-level coefficients and corresponding image detail in higher resolutions.
For N>8 (upscale) we simply need to change the Se setting in SOS and image dimensions in SOF accordingly, without any further re-encoding of the compressed data stream. So the compressed data size remains the same in these cases and does not increase with resolution.
The SmartScale extension parameters in the compressed file tell the decoder (djpeg) to take the appropriate action and output the image in the desired resolution without giving extra options.

6 Frame Offset definition in or in addition to the SOF marker segment

What we also need in a future JPEG update is a Frame Offset definition:

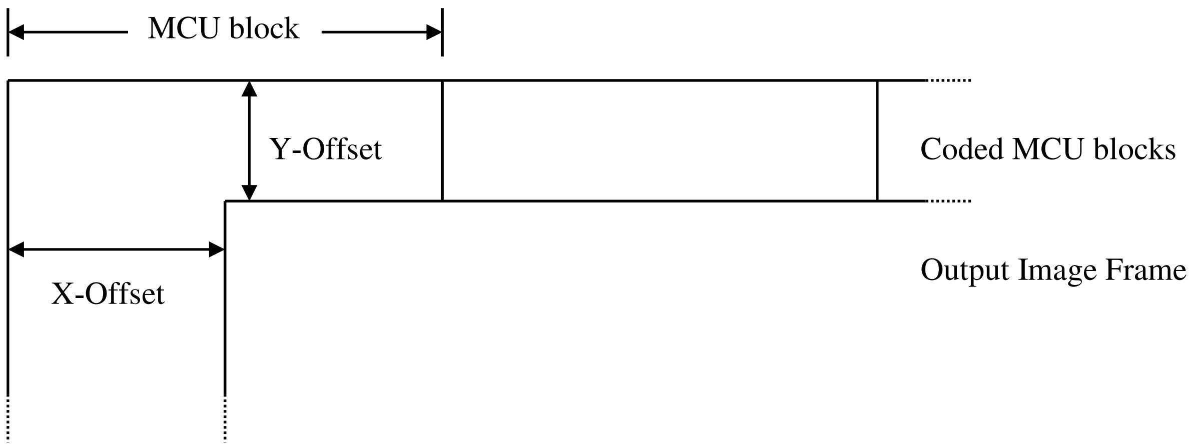


Figure 6-1 – Frame Offset definition

So the output image frame can start at an arbitrary pixel offset within the upper left coded MCU block.
This would allow arbitrary lossless croppings without any constraints for block boundaries.

Furthermore, all the lossless transformation functions (90 degree rotation etc.) could be implemented in such a way that they would work perfectly without any constraints for block-aligned image dimensions, thus without any strange edge effects and without any need for trimming options.
The author has developed the implementation of the lossless transformation and cropping functions in the IJG jpegtran utility and had and still has considerable effort to explain people the lossless transformation and cropping constraints, and these are also difficult to manage for application software developers (that's why the author has written a dedicated JPEGcrop application which is commonly used, because he knew it would be difficult for anybody else).
So it would be a big help if these functions could be provided in a way that they just work in any condition without any unnecessary constraint.
An option to define such Frame Offset in the file header would easily solve this problem. It would transfer all the effort for special handling from the user and application developer into the library, so if it is implemented correctly once there, it can be easily reused.
The Frame Offset could be easily defined in an optional JPEG marker or in an extended Start-Of-Frame (SOF) marker.

We propose here a concrete specification of the Frame Offset definition by extension of the SOF marker. This proposal has the advantage that no extra marker is required and that it is easy to adopt for existing implementations.

The frame header syntax is specified in Section B.2.2 "Frame header syntax" in T.81 as follows (Figure B.3 and Table B.2):

frame header

SOF_n	Lf	P	Y	X	Nf	component-spec. parameters

Figure 6-2 – Frame header syntax per T.81

We are not interested here in the further component-spec. parameters format.
The essential property is that the frame header consists of a fixed-size and a variable-size segment.
The Lf parameter contains the total size of the frame header segment and is specified as Lf = 8 + 3 * Nf per T.81 Table B.2. So the length of the variable size segment is determined by the parameter Nf (number of components).
We can now add additional parameters at the end of the frame header by specifying a different length value Lf.
For the JPEG-Plus Frame Offset definition we specify Lf = 10 + 3 * Nf and add two bytes at the end of the frame header segment after the variable component-spec. parameters part:

frame header

SOF_n	Lf	P	Y	X	Nf	component-spec. parameters	FOX	FOY

Figure 6-3 – Extended frame header syntax with Frame Offset definition

FOX is Frame-Offset-X, FOY is Frame-Offset-Y, each is 8 bits in size (values 0-255).

The Frame Offset values shall be smaller than the corresponding MCU dimensions, so the reserved range should suffice.
A JPEG-Plus encoder (including transcoder) may write an extended frame header per Figure 6-3 with Lf = 10 + 3 * Nf if FOX>0 or FOY>0. In case of FOX=0 and FOY=0 the encoder may choose to write a new or an old frame header.
A JPEG-Plus decoder reads first the fixed-size part of the frame header segment, then examines if Lf = 8 + 3 * Nf (old header) or Lf = 10 + 3 * Nf (new header). In case of an old header the decoder assumes FOX=FOY=0.

A Frame Offset supporting implementation must take care to adapt internal MCU related data structures appropriately. It must use values X + FOX and Y + FOY instead of X and Y previously for correct determination of right and lower block array edges. All internal processing can be done similarly with values X + FOX and Y + FOY instead of X and Y previously, only the final spatial sample output requires to regard the Frame Offset values.

Annex A Direct DCT Scaling

As a fundamental reason for using the DCT in image coding usually its "statistical decorrelation" or "energy compaction" property is given.
But thereby a fundamental property is overlooked which was apparently not yet recognized, although it appears obvious and evident afterwards:
Each nxn (n=1…8) sub-block of the 8x8 DCT can be directly interpreted as a reduced to n/8 of the linear resolution image version. The image samples of the reduced image version are obtained directly as a result of application of an nxn IDCT according to Formula (A-2) with N=n on the original 8x8 DCT coefficients calculated according to Formula (A-1) with N=8.

1-D FDCT (Forward DCT):

$$S(u) = C(u)\sum_{x=0}^{N-1} s(x)\cos\frac{(2x+1)u\pi}{2N} \qquad \text{(A-1)}$$

where

$$C(u) = \frac{1}{\sqrt{N}} \quad for \quad u = 0$$

$$C(u) = \sqrt{\frac{2}{N}} \quad for \quad u > 0$$

$$s(x) = 1 - D\ sample\ value$$

$$S(u) = 1 - D\ DCT - coefficient$$

1-D IDCT (Inverse DCT):

$$s(x) = \sum_{u=0}^{N-1} C(u)S(u)\cos\frac{(2x+1)u\pi}{2N} \qquad \text{(A-2)}$$

This very useful property is depicted in Figure A-1. The DCT block is therefore aligned with the DC coefficient in the lower left corner. Every spectral band with a certain horizontal or vertical frequency adds precisely the details of the corresponding resolution level. In other words, the DCT represents a direct separation of the image into a sequence with stepwise progressive resolution ("image pyramid").
Such property is usually associated as advantage with techniques like "Subband Coding" or "Wavelet". But while with Subband Coding or Wavelet only a resolution separation in *geometric* progression is obtained, with the DCT we achieve a finer *arithmetic* resolution progression, that is details are added in additive step sequence (1/8 image size) instead of multiplicative step sequence (usually with factor 2).

This result is summarized in Table A-1, and it also explains the relation between quantization in the DCT domain and resolution in the spatial domain: By coarser quantization of higher-level coefficients we remove detail in the corresponding resolution level. Or vice versa: If we set all coefficients *outside* the 4x4 DCT sub-block to zero (coarsest quantization), we can still reconstruct *exactly* an image with half resolution, since only the details of higher resolution levels were removed. The zeroing (special form of quantization) of higher-level frequency bands (see Figure A-1) corresponds directly to a downsampling in the spatial domain.

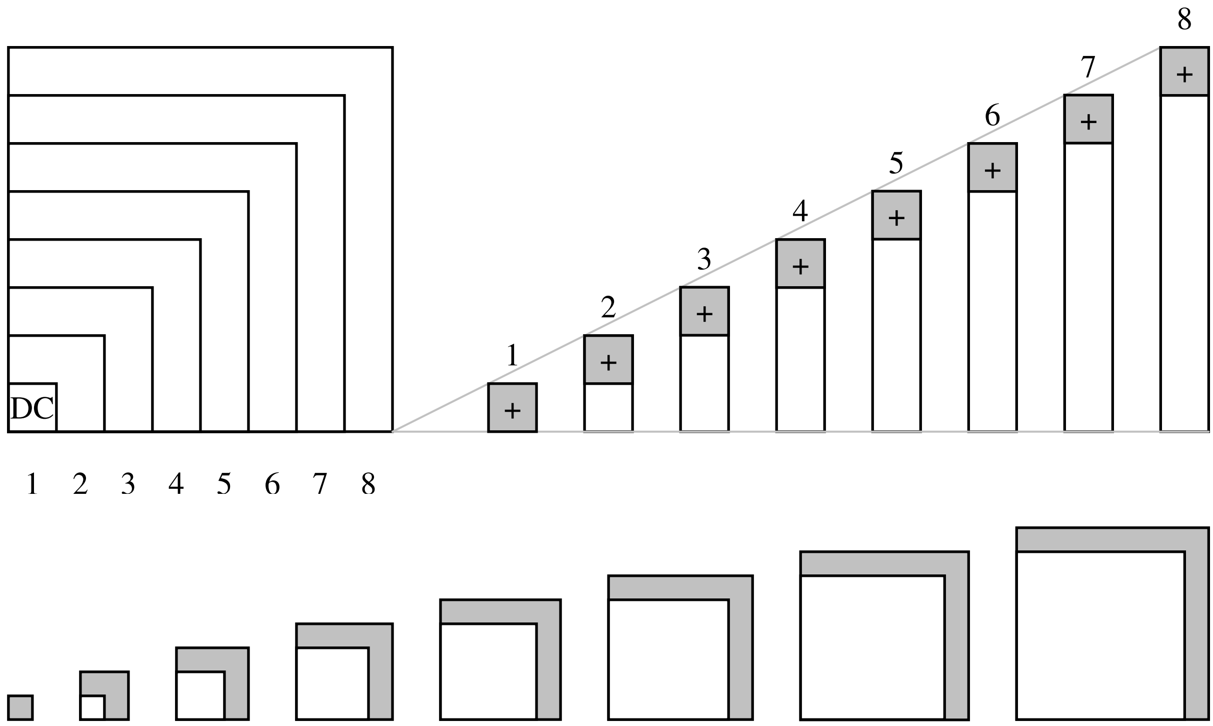


Figure A-1 – DCT block resolution bands and additive resolution progression

Table A-1 – Properties of image transformations

DCT :	Resolution separation with ***arithmetic*** progression
Subband Coding / Wavelet :	Resolution separation with ***geometric*** progression

Figure A-2 – Resolution separation in geometric progression – Property of many image coding techniques.

Figure A-3 – Resolution separation in arithmetic progression – Property of DCT based image coding.
The geometric progression appears as a subset in the arithmetic progression (framed images).
Inbetween are further resolution levels in the arithmetic sequence.

The direct DCT scaling method as described above can be extended in another direction to derive upscaled images from 8x8 DCT coefficient blocks by applying a partial higher-order NxN IDCT on the 8 input coefficients as lower frequencies and higher frequencies assumed to be zero.
It turns out that the computational effort is similar to the 8x8 IDCT regarding the output size.
For practical application it is appropriate to implement the direct DCT scaling method up to N=16 (scale factor 16/8=2), so all image scaling factors in the range N/8 with N=1…16 are supported as a decompression option, and the N=16 case (upscale factor 2) can also be used for resolving the internal color subsampling while normal decoding.

A complementary DCT scaling option can be applied in the encoding process, where we get scaling support for reverse factors 8/N with all N=1…16.
For N<8 we fill the remaining block coefficients with zero.
For N>8 we apply a partial N-point FDCT on the input samples, computing just the lower 8 frequency coefficients and discarding the rest.

The resulting NxN (N=1…16) IDCT and FDCT algorithms are implemented in a highly optimized way and are very efficient compared with the standard 8x8 size.
The following table shows the optimization results for the NxN IDCT algorithms. The FDCT results are similar with slight deviations.
We estimate the computational efficiency of the developed algorithms by the number of multiplictions required per output pixel. We do not take into account the dequantization multiplications which would improve the upscaling results (N>8) even further because only input values (less than output in case of upscaling) need to be dequantized.

Table A-1 – Computational efficiency of NxN IDCT output from 8x8 DCT coefficients algorithms

N	**1-D kernel loops**			**multiplications in 1-D kernel**			**multiplications per output pixel**
	column	**row**	**total**	**even part**	**odd part**	**total**	**= total loops * total mults / (N*N)**
1	1	1	1	0	0	0	**0**
2	2	2	4	0	0	0	**0**
3	3	3	6	1	1	2	**1.3**
4	4	4	8	0	3	3	**1.5**
5	5	5	10	2	3	5	**2**
6	6	6	12	2	1	3	**1**
7	7	7	14	7	5	12	**3.4**
8	8	8	16	3	9	12	**3**
9	8	9	17	5	5	10	**2.1**
10	8	10	18	5	7	12	**2.2**
11	8	11	19	11	13	24	**3.8**
12	8	12	20	2	13	15	**2.1**
13	8	13	21	13	16	29	**3.6**
14	8	14	22	8	12	20	**2.2**
15	8	15	23	9	13	22	**2.2**
16	8	16	24	8	20	28	**2.6**

Only the N<=8 cases are full IDCTs, the others are partial IDCTs. It is interesting that under this condition all algorithms are in a close efficiency range, regarding the output size.
The most expensive cases with slightly larger, but still very good, rating compared to the 8-point case are the prime numbers 7, 11, and 13. It seems that the even part multiplications cannot be reduced below N in these cases, while all other are smaller.
The other cases with N>8 have even slightly better ratings than the standard 8-point case.
The cases below 7 have considerably better ratings than the standard 8-point case.
Especially remarkable is the N=6 case, which has just 3 multiplications in the 1-D kernel, and a rating of 1 multiplication per output pixel.

The numbers are result of a considerable optimization effort.
Additional performance advantages are obtained by adapted band-limited entropy decoding.
This gives noticeable additional performance advantages especially for the small N values (<=4) where only few AC coefficients need to be fully decoded from the input bitstream, while the rest can be skipped.

Annex B The fundamental DCT property for image representation

Newsgroup: comp.dsp
Subject: why DCT
Date: 4 Mai 2005 13:37

Neo wrote:

> guido,
> I found out that the DCT distributes most of the energy in the lower
> order co-efficients compared to the FFT and also gives a purely real
> output. This energy localisation aids in more efficient encoding later
> on.
> Is this reasonable?

No!;-)
Your question was why is the DCT used in *image* compression!
The general term "energy compaction" does not explain why it is good
just for *image* compression, you could say the same thing for any
other compression object.

The actual reason is much simpler, and therefore apparently very
difficult to recognize by complicated-thinking people.

Here is the explanation:

What are people doing when they have a bunch of images and want
a quick preview? They use thumbnails! What are thumbnails?
Thumbnails are small downscaled versions of the original image!
If you want more details of the image, you can zoom in stepwise
by enlarging (upscaling) the image.
That is the key to understanding the use of DCT for image compression:

The fundamental property of lossy image compression is the similarity
of different resolutions of the same image. "Lossy" compression means
that we assign *the same* output representation to *multiple*, *similar*
input representations. The basic similarity relation for images is
resolution, or scale, invariance: If we see the same image in different
resolutions (scales, sizes), or the same subject from different distances,
we talk about *the same* image (or subject).

The DCT provides the best resolution separation property for digital images.
The 8-point DCT gives you 8 linearly increasing resolution representations
from 8 spatial sample values. You can hardly do better than that.
Wavelet transforms, as used in JPEG2000, for example, do *not*
provide such optimal resolution separation.
See also chapter 4 of my paper at http://jpegclub.org/temp/.

Everybody who knows the DCT knows that the DC term represents a
1/8 scale of the input sequence ("thumbnail" version).

The DC and first AC together represent a 2/8 or 1/4 scale of the input sequence. The DC and first 2 ACs together represent a 3/8 scale of the input sequence, and so on.
Every DCT coefficient adds corresponding resolution detail.
This is easy to demonstrate, but was not known before.
(See new JPEG scaling features presented at http://jpegclub.org which directly apply this property.)
This fundamental DCT property explains why the DCT is the best transform for image compression.

Regards
Guido

Figure B-1 – Similarity of different resolution (zoom) levels of the same image as fundamental property for image coding