The NR JPEG2000-codec

Norsk Regnesentral
ANVENDT DATAFORSKNING

Norwegian Computing Center/Applied Research and Development

NOTAT/NOTE

DART/08/04

Hans Jakob Rivertz
Svetlana Boudko

Oslo
Høst 2004

Norsk Regnesentral
ANVENDT DATAFORSKNING

Notat / Note

Tittel/Title:
The NR JPEG2000-codec

Dato/Date: April
År/Year: 2004
Notat nr/:
Note no: DART/–/04

Forfatter/Author:
Svetlana Boudko,
Hans Jakob Rivertz

Sammendrag/Abstract:

In the ComDec project we have implemented codecs for JPEG2000 and MJPEG2000. This report describes the status of our work with the codec. The project has been a part of the ChannelS strategic institute program (SIP). The codec we developed can compete with most of the JPEG2000 codecs that are available. The codec can compete in quality with other JPEG2000 codecs.

Emneord/Keywords: JPEG2000, codec

Målgruppe/Target group: <To be inserted>

Tilgjengelighet/Availability: Open

Prosjektdata/Project data: COMDEC

Prosjektnr/Project no: 11008

Antall sider/No of pages: 19

Norsk Regnesentral/Norwegian Computing Center
Gaustadalléen 23, Postboks 114, N-0314 Oslo, Norway
Telefon (+47) 22 85 25 00, telefax (+47) 22 69 76 60

Contents

Chapter 1

Implementation of the JPEG 2000 Standard

ComDec is a sub project of the ChannelS[?] Strategic Institute Program. Its goal was develop a JPEG2000 codec. This report describes the results / implementations developed in the project.

1.1 Introduction

In this project we have implemented an encoder and a decoder for the JPEG2000 standard [?] and an encoder and a decoder for MJPEG2000 [?]. The developed codecs compile and run on several platforms, e.g. Microsoft Windows, Linux, Unix, and iTron[?]. The participants in the project have been Sandor Seres, Wolfgang Leister, Eirik Maus, Lars Aarhus, Svetlana Boudko, Per Røe, and Hans Jakob Rivertz.

Throughout the project we have emphasized on implementation issues of the JPEG2000 codec, platform independence, demonstrators and new ideas for the further work.In the autumn 2003 NR developed a codec for the part 3 of the JPEG2000 standard, namely MJPEG2000 which is a standard for packing frames in moving pictures encoded with JPEG2000. Our implementation supports all obligatory parts of part 3 of the JPEG2000 standard.

1.2 The JPEG2000 codec

1.2.1 Requirements

Our requirement specification for a successful implementation included the following features of the JPEG2000 standard.

- Lossy and Lossless compression. The JPEG2000 standard supports lossy and lossless compression. Lossy compression can be done in two ways. Either by explicit quantization or implicit by truncating the compressed file.
- Arbitrary Tile offset.
- As input to the codec, RGB with 8 bit per channel is required as a minimum.
- Arbitrary Tile Size. Compressing big images, (e.g. several mega pixels or giga pixels,) requires a division of images into smaller parts, (tiles).
- One precinct. Precincts is the highest level of spatial organization of tiles in a picture.
- Many quality layers. The JPEG2000 standard offer a flexible organization of image information. Quality layers are parts of the file that gives
- Quality control.
- At least one progression: Layer — resolution level — color component — precinct shall be supported. Progressions are ways of organizing the information in the file such that the most

important information is coming before the less important information. What is important vary in applications and therefore there are more than one progression in the JP standard.
- The obligatory markers in annex A and the obligatory boxes in annex I of first part of the JPEG2000 standard [?]

1.2.2 Status

After implementing the JPEG2000 codec we tested it with the Kakadu codec [?] as a reference. We used the Kakadu codec because bench tests showed that it is the fastest software based JPEG2000 codec available and it is also well acknowledged. Our encoder does support all features mentioned in section 1.2.1. Our implementation competes well in performance compared with other encoders. Refer to Appendix A.6 for test results.

To implement the JPEG2000 decoder was simpler than to implement the encoder. A lot of code that had been used in the encoder could be reused in the decoder. The code that had to be implemented separately was often very similar to corresponding code in the encoder. Most of the data structure that was written for the encoder could be used in the decoder.

1.3 The MJPEG2000 codec

1.3.1 Requirements

Our requirement specification necessary to implement a MJPEG2000 (moving JPEG2000) codec included the following list

- Selection of the format to use for the input stream for the MJPEG2000 encoder.
- Reuse code from the JPEG2000 encoder.
- Implement a MJPEG2000 codec
  - Implement an MJPEG2000 file structure parser and writer. A minimal implementation should include all that are obligatory in the MJPEG2000 standard.
  - Integrate our JPEG2000 codec into the MJPEG2000 codec.
- Compare our codec with existing codecs.
- Implement modules for mplayer or Microsoft Media for demonstration purposes.

1.3.2 Status

MJPEG2000 is to JPEG2000 as MJPEG is to JPEG, namely a moving picture format, where all frames are I-frames. We use QuickTime™[1] as input file format since the MJPEG2000 format is based on the QuickTime format. Therefor we could reuse a lot of the code. The MJPEG2000 file format is build up by 'boxes'. The MJPEG2000 file format parser and reader were implemented to support all the obligatory boxes in the format.

This saved us for a lot of work. Some of the modules, such as the file format parser and block header parsers were already made in for debug reason when we implemented the JPEG2000 encoder.) We used the Morgan Media MJPEG2000 codec [?] to test the correctness of our codec. Our files played without problems on the Morgan decoder. Modules for mplayer and Microsoft Media are planned.

[1] QuickTime is a trademark own by Apple Computer

Chapter 2

The porting to iTron

Many Japanese cameras use operating systems that are based on the specification iTron^{TM}[?]. NR have worked with a test device for software development manufactured by T-Engine [?]. We were among the first outside japan that implemented application running on this iTron.

NR believes that our competence could be important for the camera marked segment. Software will be faster on small devices in the future. Therefore software based codecs should be more common. For preparation we decided to port our JPEG2000 codec on the iTron^{TM}platform. We have made demonstration applications for this iTron device.

2.1 Requirements

The main goal was to port the JPEG2000 codec and the MJPEG2000 codec to the iTron platform.

- Install the iTron development kit (Hard- and Software)
- Install the development tool on a computer with Linux OS.
- Port the JPEG2000 codec module to iTron
- Port the MJPEG2000 codec module to iTron
- Implement demonstrators for viewing JPEG2000 still images and MJPEG2000 movies on this device

2.2 Status

We implemented the JPEG2000 codec on the iTron device. The MJPEG2000 codec could not be ported to the device since the code of this codec was written i C++ and the develop kit did not support C++ at the time of the deadline. We chose to make a video player that played concatenated JPEG2000 pictures. [1]

We implemented the following applications:

jp2dec A JPEG2000 decoder that decodes a j2k or jp2 file and write a tiff file.

jp2enc A JPEG2000 encoder that makes a j2k or jp2 file from a tiff file.

jp2show A JPEG2000 viewer of small j2k / jp2 files.

mjp2player A JPEG2000 sequence viewer.

A power point presentation of how to port to iTron is made and can be found at:
"`/nr/group/dart/Publikasjoner/Populaer/porting til ITRON.ppt`"
"`G:\dart\Publikasjoner\Populaer\porting til ITRON.ppt`"

[1] T-Engine does now support C++ so this porting job now is possible now.

Appendix A

Technical

A.1 Other Implementations

There exist several implementations of JPEG2000 in the market. The quality of these implementations is varying. We have made measurements of how the different implementations performed and visually compared the quality of lossy compressed images.

The conclusion is that only the Kakadu implementation can compete with our implementation when it comes to quality and CPU usage.

A.2 Detailed description of the JPEG 2000 Implementation on NR

This section contains a documentation on which parts that were implemented in our codec in the end of June 2004.

The following table shows the symbols used in the tables of this section

- **x**: Implemented
- **p**: Partly implemented
- **n**: Not implemented
- **i**: Implemented but should be improved
- **-**: Not allowed

A.2.1 Transformations

We support all wavelet filter banks in part I of the standard. The following table indicate the status for each of the implemented filter bank.

Feature	Decoder	Encoder
9-7 Wavelet	**i**	**i**
5-3 Wavelet	**x**	**x**
ICT	**i**	**i**
RCT	**x**	**x**

A.2.2 Markers

Most of the markers in part I of the JPEG2000 standard are implemented. The table shows all markers in part I of the standard and their status in our implementation.

Marker	Symbol	Decoder		Encoder	
		MainHeader	TileHeader	MainHeader	TileHeader
Start of code stream	SOC	**x**	**-**	**x**	**-**
Start of tile part	SOT	**-**	**x**	**-**	**x**
Start of data	SOD	**-**	**x**	**-**	**x**
End of code stream	EOC	**-**	**-**	**-**	**-**
Image and tile size	SIZ	**x**	**-**	**x**	**-**
Coding Style Default	COD	**x**	**x**	**x**	**x**
Coding Style Component	COC	**x**	**x**	**n**	**n**
Region Of Interest	RGN	**x**	**x**	**n**	**n**
Quantization Default	QCD	**x**	**x**	**x**	**p**
Quantization Component	QCC	**x**	**x**	**n**	**n**
Progression order change	POC	**p**	**p**	**n**	**n**
Tile part length	TLM	**x**	**-**	**p**	**-**
Packet length main header	PLM	**n**	**-**	**n**	**-**
Packet length tile-part header	PLT	**-**	**n**	**-**	**n**
Packed packet header, main header	PPM	**n**	**-**	**n**	**-**
Packed packet header, tile-part header	PPT	**-**	**n**	**-**	**n**
Start of Packet	SOP	**-**	**x**	**-**	**x**
End of packet header	EPH	**p**	**x**	**p**	**x**
Component registration	CRG	**x**	**-**	**n**	**-**
Comment	COM	**x**	**x**	**n**	**n**

A.2.3 Status of features of the implementation

The following table contains features of the JPEG2000 standard and the status of our implementation.

Feature	Decoder	Encoder
Precincts	**x**	**p**
Use SOP	**x**	**x**
Use EPH	**x**	**x**
Prog order	**i**	**i**
Num of layers	**x**	**x**
Code block width	4-1024 coefficients	4-64 coefficients
Code block height	4-1024 coefficients	4-1024 coefficients
Transformations	5-3 and 9-7	5-3 and 9-7
Precinct Size	All	One
Layer-res-comp-pos order	**x**	**x**
Res-layer-comp-pos order	**x**	**x**
Res-pos-comp-layer order	**i**	**n**
Pos-comp-res-layer order	**i**	**n**
Comp-pos-res-layer order	**i**	**n**
Bypass	**x**	**n**
Reset probability	**x**	**x**
Pass termination	**x**	**n**
Vertical context	**x**	**x**
Predicted termination	**x**	**x**
Segmentation symbols	**x**	**x**
Region of interest	**i**	**n**
Quantization	**i**	**i**
Output input Bit depths	8	8
Gray scale	**x**	**x**
RGB-color	**x**	**x**
CMYK-color	**x**	**x**
Palette	**x**	**n**
YUV 4:2:2	**x**	**n**
YUV 4:1:0	**x**	**n**

A.2.4 Supported input file formats

The following input formats are supported

- encoder: uncompressed Tiff, ppm, mov(RGB)
- decoder: j2k, jp2, mj2

A.2.5 Supported output file formats

The following output formats are supported

- encoder: j2k, jp2, mj2(RGB)
- decoder: uncompressed Tiff, mov

A.2.6 JPEG 2000 Boxes

The following table contains a list of those jp2 file format boxes that our codec support. The mentioned boxes are all those which is defined in the JPEG2000 standard.

Name of Box	Type	decoder	encoder
JPEG2000 Signature Box	'jP '	**x**	**x**
File Type Box	'ftyp'	**x**	**x**
JP2 Header Box	'jp2h'	**x**	**x**
Image Header Box	'ihdr'	**x**	**x**
Bits Per Component Box	'bpcc'	**x**	**n**
Color Specification Box	'colr'	**x**	**x**
Palette Box	'pclr'	**x**	**n**
Component Mapping Box	'cmap'	**x**	**n**
Channel Definition Box	'cdef'	**x**	**x**
Resolution Box	'res '	**x**	**x**
Capture Resolution Box	'resc'	**x**	**x**
Default Display Resolution Box	'resd'	**x**	**x**
Contiguous Code stream Box	'jp2c'	**x**	**x**
Intellectual Property Box	'jp2i'	**n**	**n**
XML box	'xml '	**x**	**n**
UUID Box	'uuid'	**n**	**n**
UUID Info Box	'uinf'	**n**	**n**
UUID List Box	'ulst'	**n**	**n**
URL box	'url '	**n**	**n**

A.2.7 MJPEG2000 boxes

At least all obligatory boxes of the MJPEG2000 standard is implemented.

Box name	Description	Parent box	MJPEG2000 Encoder	MJPEG2000 Decoder
'jP '	Jp2 signature		**x**	**x**
'ftyp'	File type		**x**	**x**
'mdat'	Media data		**p**	**p**
'moov'	Container for all meta-data		**p**	**p**
'mvhd'	Movie header	moov	**x**	**x**
'trak'	Container for a track	moov	**x**	**x**
'tkhd'	Track header	trak	**x**	**x**
'tref'	Track reference container	trak	**x**	**x**
'edts'	Edit list container	trak	**n**	**n**
'elst'	An edit list	edts	**n**	**n**
'mdia'	Container for media information in a track	trak	**x**	**x**
'mdhd'	Media header	mdia	**x**	**x**
'hdlr'	Handler for media type	mdia	**x**	**x**
'minf'	Media information container	mdia	**x**	**x**
'vmhd'	Video media header	minf	**x**	**x**
'smhd'	Sound media header	minf	**x**	**x**
'hmhd'	Hint media header	minf	**p**	**p**
'dinf'	Data information box	minf	**x**	**x**
'dref'	Data reference box	dinf	**p**	**p**
'stbl'	Sample table box minf		**x**	**x**
'stsd'	Sample descriptions	stbl	**p**	**p**
'stts'	Time to sample	stbl	**x**	**x**
'stsc'	Sample to chunk	stbl	**x**	**x**
'stsz'	Sample sizes	stbl	**x**	**x**
'stco'	Chunk offset	stbl	**x**	**x**
'mvex'	Movie extends box	moov	**n**	**n**
'trex'	Track extends default	mvex	**n**	**n**
'moof'	Movie fragment		**n**	**n**
'mfhd'	Movie fragment header	moof	**n**	**n**
'traf'	Track fragment	moof	**n**	**n**
'tfhd'	Track fragment header	traf	**n**	**n**
'trun'	Track fragment run	traf	**n**	**n**
'free'	Free space		**x**	**x**
'skip'	Free space		**x**	**x**
'udta'	User-data, copyright	moov	**x**	**x**

A.2.8 The structure

The framework contains the following files:

Main files

File name	JPEG-2000 encoder	JPEG-2000 decoder	MJPEG-2000 encoder	MJPEG-2000 decoder
jp2totiff.cpp		+		
ppmtojp2.cpp	+			
tifftojp2.cpp	+			
MPEG2000decoder.cpp				+
MPEG2000encoder.cpp			+	

Source files

File name	JPEG-2000 encoder	JPEG-2000 decoder	MJPEG-2000 encoder	MJPEG-2000 decoder
blockBuffer.cpp	+	+	+	+
boxbuffer.cpp			+	+
coeffbitmodel.cpp	+	+	+	+
ComDec.cpp	+	+	+	+
counter.cpp	+	+	+	+
daubechies.cpp	+	+	+	+
decodeBlock.cpp		+	+	+
decoder.cpp		+		+
decodeTile.cpp		+		+
encodeBlock.cpp	+		+	
encoder.cpp	+		+	
encodeTile.cpp	+		+	
entropyCodec.cpp	+	+	+	+
j2k_headers.cpp	+		+	
jp2_headers.cpp	+		+	
jp2packet.cpp	+		+	
lazy.cpp	+		+	
mainboxes.cpp			+	+
parsej2k.cpp		+		+
parsejp2.cpp		+		+
parsemjpeg2.cpp			+	+
qtatom2.cpp			+	+
qtatoms.cpp			+	+
resolution.cpp	+	+	+	+
tifftools.cpp	+	+	+	+
tile.cpp	+	+	+	+

Header files

File name	JPEG-2000 encoder	JPEG-2000 decoder	MJPEG-2000 encoder	MJPEG-2000 decoder
blockBuffer.h	+	+	+	+
codeword.h		+	+	+
coeff.h	+	+	+	+
coeffbitmodel.h	+	+	+	+
ComDec.h	+	+	+	+
ComDec_defs.h	+	+	+	+
ComDec_funcs.h	+	+	+	+
contex.h	+	+	+	+
counter.h	+	+	+	+
daubechies.h	+	+	+	+
debug.h	+	+	+	+
decodeBlock.h		+		+
decoder.h		+		+
decodeTile.h		+		+
encodeBlock.h	+		+	
encoder.h	+		+	
encodeTile.h	+		+	
entropyCodec.h	+	+	+	+
j2k_headers.h	+		+	
jp2_headers.h	+		+	
jp2packet.h	+		+	
lazy.h	+	+	+	+
parsejp2.h		+		+
resolution.h	+	+	+	+
RGBtiffIFD.h	+	+	+	+
tifftools.h	+	+	+	+
tile.h	+	+	+	+
boxbuffer.h			+	+
boxsize.h			+	+
boxtype.h			+	+
constsizevar.h			+	+
ERRORMSG.h			+	+
inttype.h			+	+
parsempeg2.h			+	+
putgetproc.h			+	+
qtatom2.h			+	+
qtatoms.h			+	+
winfiles.h			+	+

A.3 Conformance

The part 4 of the JPEG2000 standard [**?**] describes conformance testes. There are 3 profiles and we have tested 2 of them profile 0 and profile 1. The test consists of a number of JPEG2000-encoded pictures and the corresponding uncompressed files. The following tables show how far we have developed our decoder.

A.3.1 J2K-conformance profile 0 test results

On march 22 2004, our decoder was tested for conformance. Here is the result of that test.

FILE	Status decoding	Status wavelet	Status out file	Comments
p0_01.j2k	OK	OK	OK	"Plain" j2k file
p0_02.j2k	OK	OK	False resolution ratio and size	
p0_03.j2k	FAILS	-		
p0_04.j2k	OK	OK	OK	
p0_05.j2k	OK	OK		False resolution ratio and size of CMYK components
p0_06.j2k	FAILS			(problem related to too big packets.)12 bit+RGN in main and tile
p0_07.j2k	FAILS			12 bit Large number of tiles
p0_08.j2k	FAILS			12 bit Large image
p0_09.j2k	OK	OK	OK	Overflow in IWT 9-7
p0_10.j2k	FAILS			
p0_11.j2k	OK	OK	OK	sample high
p0_12.j2k	OK	OK	OK	Very small image
p0_13.j2k	FAILS			
p0_14.j2k	OK	OK	OK	
p0_15.j2k	FAILS			
p0_16.j2k	OK	OK	OK	Empty packet header bit

A.3.2 J2K-conformance profile 1 test results

On march 22, the pack&do decoder was tested for conformance. Here is the result of that test.

FILE	Status decoding	Status wavelet	Status out file	Comments
p1_01.j2k	OK	OK		False resolution ratio and size
p1_02.j2k	FAILS			
p1_03.j2k	FAILS			
p1_04.j2k	FAILS			
p1_05.j2k	FAILS			
p1_06.j2k	FAILS			
p1_07.j2k	FAILS			

A.3.3 Jp2-conformance test results

On march 22, the pack&do decoder was tested for conformance. Here is the result of that test.

FILE	Status decoding	Status wavelet	Status out file	Comments
file1.jp2	OK	OK	OK	24 bit RGB-image
file2.jp2	OK	OK	Color misinterpretation	
file3.jp2	OK	OK	OK	24 bit YCC - image with sub-sampling
file4.jp2	OK	OK	OK	8 bit Gray scale image
file5.jp2	OK	OK	ICC missing,or ROMM RGB color space indicated not supported.	
file6.jp2	OK	OK	Scale misinterpretation	12 bit Gray scale image
file7.jp2	OK	OK	Color misinterpretation	48 bit YCC - image
file8.jp2	OK	OK	ICC missing	8 bit Gray scale image with ICC profile
file9.jp2	OK	OK	OK	Palette image

A.4 Executables

There are four executables:

Executable	Name of executable
Encoder for JPEG2000 format	tifftojp2(.exe)
Decoder for JPEG2000 format	jp2totiff(.exe)
Encoder for MJPEG2000 format	MPEG2000encoder(.exe)
Decoder for MJPEG2000 format	MPEG2000Parser(.exe)

A.5 Supported Platforms

The implementation of JPEG2000 encoder/decoder has been verified to run on the following platforms: (Continuously tested during development)

- Linux on Intel
- Windows
- Tron

(Less frequently tested)

- Pocket PC (MIPS)
- Embedded Linux (Power PC)
- Unix (Solaris / Sparc)

The implementation of MJPEG2000 encoder/decoder has been verified to run on the following platforms:

- Linux on Intel
- Windows
- Unix(Solaris)

A.6 Comparison of the NR codec and the KAKADU codec

A.6.1 Test Computer

Two Intel(R) Xeon(TM) CPU 2.80GHz

CPU MHz 2799.348

Cache size 512 KB

Flags fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm

A.6.2 Comparison for lossless compression. KAKADU versus NR.

The Kakadu encoder used has version number 4.3. Both codec encodes with 4 layers and lossless. The user CPU time were measured to

Filename	Size	NR encoder	kdu compress	NR decoder	kdu expand
jenny	1800x1200	1.16	1.28	1.37	1.32
Sveta_working800	800x600	0.31	0.34	0.39	0.36
mini	300x300	0.21	0.26	0.24	0.23