The Dirac video coding system

Thomas Davies and Anuradha Suraparaju, BBC Research and Development

1 Introduction

In March 2004, BBC Research and Development released the Dirac video codec as open source software [1]. Since being released, Dirac has continued to improve and now provides compression performance comparable with SMPTE VC-1 and MPEG H264.

The philosophy behind Dirac is 'keep it simple'. Dirac contains a small number of core tools, and obtains its compression performance by using tools with good subjective performance and optimising the tools well, our aim being to avoid the great complexity of other compression algorithms. It is also our aim to provide copious documentation, of the algorithm, the code, the API and (ultimately) the bitstream syntax so that Dirac can be used in as many applications as possible, and also used as a framework for future codec development in the free and open source domain.

This paper provides an introduction to the Dirac technologies and software, and describes where we see Dirac going in the next few months.

2 Dirac architecture and coding tools

Dirac's overall architecture is conventional, in that it is a 'hybrid' motion-compensated codec. Hybrid, because transforms are applied both to some pictures themselves (Intra frames), but also to frames which have been predicted from previously coded frames (Inter frames). However, how Dirac does transform coding and prediction is different from other codecs (full details of the algorithm are available on the homepage [2]).

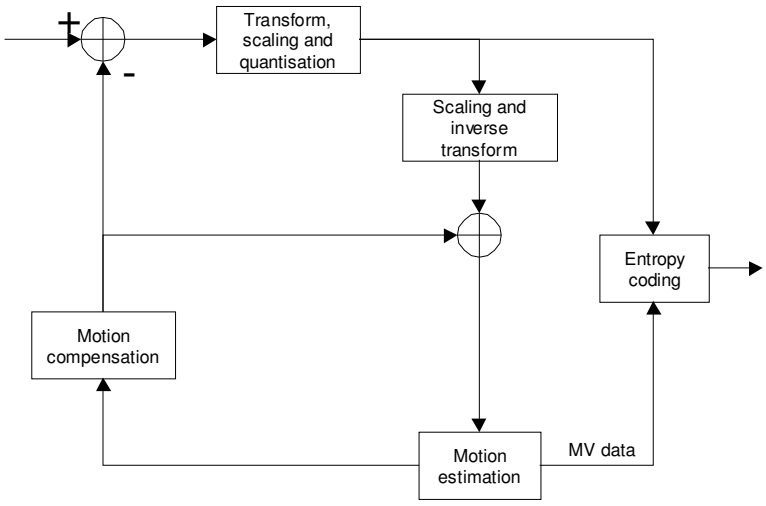

Figure 1: Dirac encoder architecture. The decoder performs the inverse operations

2.1 Transform

Dirac's transform is not a block transform, but instead is a wavelet transform applied to the whole picture. Wavelets provide a multi-resolution representation of an image, so that it can be viewed as successively lower resolution versions of the image, together with 'difference' signals that allow the picture to be reconstructed successively (Figure 2). This property gives very good results in perceptual terms, since at low bit rates the detail contained in the difference signals can be thrown away to produce a softer image without blockiness (Figure 3).

Figure 2. Wavelet transform of 'Lena'

Figure 3. Intra coded picture at a)1 bit per pixel and b) at 0.26 bits per pixel.

2.2 Motion compensation

Motion compensation in Dirac does use blocks, but the blocks are overlapping. Across the overlaps, there is a smooth transition between the predictions provided by the different motion vectors. The result is that even at low bit-rates Dirac is generally free from blockiness for inter as well as intra frames.

2.3 Encoder control

The Dirac software encoder is currently controlled by a single quality setting, which sets a target quality for the frames, and dynamically controls encoding so that video quality stays roughly constant at this target. The quality metric is a perceptual metric designed to penalise the most objectionable coding artefacts, such as large quantisation errors, image noise 'pumping' and false motion. Control is by means of a feedback loop (Figure 4) which models the relationship between coding parameters and quality to steer encoding towards the target quality. This is purely on the encoder side and does not affect the decoder at all.

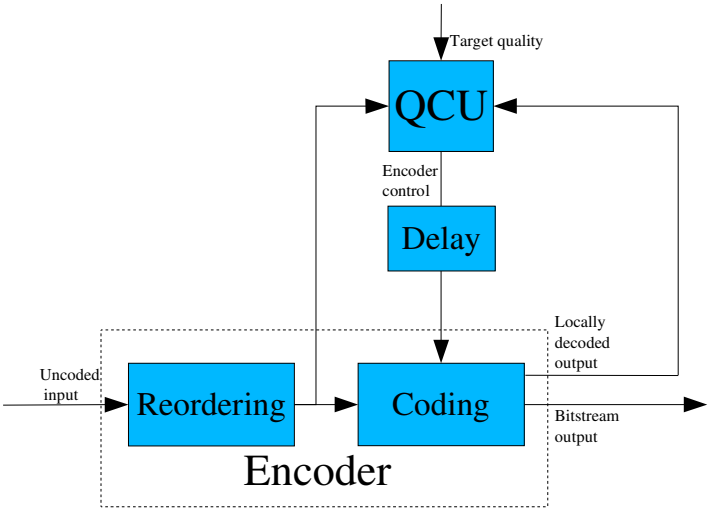

Figure 4. Controlling the encoded quality with a feedback loop

3 Dirac Software

A reference implementation of Dirac was released Open Source under the Mozilla triple licence [3] in March 2004. This licence allows the Dirac software to be used in both free and proprietary products. It also allows for relicensing under the GPL or the LGPL. The reference implementation and related documentation is hosted on Sourceforge and can be downloaded from the Dirac Project page [1].

3.1 Software Environment

Dirac software is designed to be platform independent. The source code is distributed as a compressed tarball. The codec libraries can be built from source on all POSIX conforming operating systems using ISO C++ [4] compliant compilers. This is made possible by using the GCC compiler [5] and GNU Build system [6,7]. Dirac libraries can also be built on MS Windows 2000/XP platforms using MS Visual C++ 2003. Dirac has been tested under different distributions of GNU/Linux, BSD, Solaris, IRIX, Darwin and MS Windows 2000/XP. The build produces an encoder library, a decoder library, command line driven encoder and decoder executables, and diagnostics and transcoding tools.

3.2 Input & Output Video formats

The Dirac encoder and decoder currently uses planar YUV formats. Chroma formats supported include Y only (grayscale), 4:2:0, 4:2:2 and 4:4:4.

3.3 API

The core Dirac software is implemented in ISO C++. Since codec implementation can be very complex, it was decided that the Dirac library would be implemented using a language that encourages object oriented design, particularly function and data abstraction. Although the Dirac internals are written in C++, for clients of the library a “C” external API was chosen since the external API was designed to be simple as possible and a “C” interface was sufficient to achieve this goal. The “C” interface also provides straightforward integration into media players, video processing tools and streaming software.

Detailed description of the Dirac API and sample encoder/decoder code can be found at [8]. The Dirac bitstream is still evolving, so the API is still likely to change over the next few releases. But the design is such that the changes will be minimal.

The following lists the key features of the Dirac API.

3.3.1 Input driven state machine model

The current state is maintained in the encoder/decoder object. The state table is implemented as a switch in which each case defines a state. The API user evaluates the object state repeatedly and takes action depending on its value.

The basic API usage is as follows -

```
Set up the encoder/decoder object
while (true)
{
        monitor state of encoder/decoder object
        take action depending on state of object
}
Teardown the encoder/decoder object
```

3.3.2 Flexible encoder API

Encoding is a complex process that requires numerous parameters to be set to take suitable encoding decisions. The encoder API is designed so that the encoder can be used as blackbox by using presets which set the parameters to default values, or can be fined tuned by setting parameters individually.

3.3.3 Memory buffer based I/O

Memory buffers allow for easy integration into media players and streaming servers. The buffers are managed by the API users.

3.3.4 Frame based I/O

One full frame of uncompressed data must be loaded into the encoder before calling the encoding routines. The output of the encoder is one compressed frame at a time. The order of frames in the compressed bitstream is not the same as the display order of the frames.

The decoder stores the input internally until it has sufficient data to uncompress one frame successfully. It outputs frames in display order. This implies a delay in the decoded output which depends on the GOP structure. Typically there is a delay of 2 frames at the beginning of the decode process for a 'classic' MPEG GOP structure of I B B P B B P ...[1]

4 Integration into Media Players

A codec needs to satisfy certain criteria to be able to be integrated into a media player. These include the following.

4.1 Bit stream identification

The media player should be able to identify the bitstream to decide if it can be played back. Every Dirac bitstream starts with the unique sequence of 8 bytes "KW-DIRAC" in ASCII that identifies the bitstream as Dirac.

[1]Dirac frame types differ slightly from MPEG2 types, but the difference is not material here.

4.2 Structural Metadata

In order to playback, video media players need some basic information about the video sequence – e.g. video dimensions, output colour format and frame rate. The Dirac bitstream carries such structural metadata.

4.3 Output formats

Dirac outputs individual buffers for the Y, U and V frames, a format that is used by many media players.

4.4 Bit stream parsing

Dirac has unique start codes, similar to MPEG-2, to identify the start of parse units in a bitstream. In addition, although this feature is not yet supported, the bitstream specification also specifies that byte offsets to the next and previous frame are included in the frame headers, forming a doubly-linked list. This feature can be used by media players to skip frames, read a frame of data before passing it to the decoder, as well as fast-forward and rewind without actual decoding or pre-indexing the bitstream.

4.5 Suitable API

As mentioned in the earlier section, the Dirac API is memory buffer based and not file based, which makes it easy to integrate into existing media players. The API usage is simple, and support for Dirac can be added into media players with very little effort.

4.6 Video player and streaming support

The Dirac developers have patched MPlayer allowing it to play back raw Dirac bitstreams. We hope to include this patch in the Mplayer distribution in April or May this year. In the mean time it is available on the Dirac Project page [1].

Dirac video, wrapped in Ogg, is supported for playback in the VLC media player. A Dirac plug-in is also available in gstreamer.

DirectShow filters have been developed by members of the Open Source community to enable playback using Microsoft's Windows Media Player. A DirectShow filter implementation can be found at [9] or on the Dirac project page [1]. Support for Java Media Framework is planned in the future.

5 Container Formats

Dirac is a video codec and video on it own is not very useful in a media player. It needs to be accompanied by audio. For Dirac to be adopted as an alternative to other video codecs in media players, it needs to be wrapped to be synchronised with audio and possibly other data such as subtitles. There are several wrappers or container formats available today. Some of the commercially licensed formats are ASF, AVI, MPEG-PS & MPEG-TS, Quicktime and Real. Licence free formats include Ogg, Matroska and MXF.

At the time of writing, we are focusing our efforts on MPEG-2 Transport Streams and MXF for the following reasons.

MPEG2 transport streams are a mature technology and already used to carry other types of elementary streams such as WM9. There is a recommended code of practice [10] by ProMPEG, based on an RFC 2250 [11], which deals with interleaving and error correction of MPEG-2 transport streams. There do not seem to be any patent issues associated with using MPEG-2 transport streams. This choice is a suitable way of streaming Dirac.

MXF [12] is a file format optimised for interchange of material in the content creation industries. It came into existence through a strong collaboration between the end users of the format (broadcasters, production houses, etc.) and the manufacturers of broadcasting and editing equipment (through the Pro-MPEG Forum and AAF). MXF is designed to be both a file storage format and a streaming format.

These choices don't preclude the use of other wrappers. By providing a suitable and well documented API, we make it simple for Dirac to be wrapped in other container formats as well.

6 Road map for future work

6.1 Time scales and targets

Dirac is still at alpha, the most recent release (at the time of writing) being 0.5.1. This provides very good coding performance with the encoder, and the decoder frame rate is approximately 10 frames per second for standard definition pictures on a 3GHz Pentium machine. We intend to move to beta by the end of this year, with the beta release including:

- real-time software decoding of standard definition TV on moderate PC platforms
- bitstream syntax specification, together with a compliant encoder
- incorporation of new coding tools such as global motion estimation and interlace tools
- Constant Bit Rate (CBR) coding
- Bitstream error resilience

The intention is to support the development of other implementations of Dirac in hardware as well as software, by providing a specification of the bitstream and decoder semantics. Decoder semantics are unlikely to be completed by the time of the beta release as there may need to be changes to the algorithm in order to speed up decoding. Global motion estimation is currently being developed on a separate branch and should be incorporated into the main branch soon. The inclusion of interlace tools in the beta release is more speculative, as they may require significant refactoring of the code, and so they may instead be included later by specifying a new coding profile.

For the moment, we have left error detection and correction as features to be supported in whatever transport medium carries Dirac bitstreams. However, the decoder still needs to be robust against bit errors, which in software can cause the decoder to crash unexpectedly or hang indefinitely. An exception handling framework will be implemented to mitigate these effects. We are also leaving room in the bitstream syntax for an error-resilient profile incorporating FECs to be defined later.

7 References

[1] http://sourceforge.net/projects/dirac
[2] http://dirac.sourceforge.net/
[3] http://www.mozilla.org/MPL/MPL-1.1.html
[4] ISO/IEC 14882 - Programming language C++
[5] http://gcc.gnu.org/
[6] http://www.gnu.org/software/automake/
[7] http://www.gnu.org/software/autoconf/
[8] http://dirac.sourceforge.net/documentation/code/programmers_guide/index.htm
[9] http://sourceforge.net/projects/guliverkli
[10] http://62.73.167.57/publications/pdf/Vid-on-IP-CoP3-r2.pdf
[11] http://www.faqs.org/rfcs/rfc2250.html
[12] 'Material eXchange Format (MXF)', Jim Wilkinson and Bruce Devlin in 'The file interchange handbook for images, audio and metadata', Editor in Chief Brad Gilmer, Focal Press, ISBN 0240806050