Compress a RAW video

- Foreman sequence (QCIF - 176 × 144)
- Foreman sequence (CIF – 352 x 288)

1. Extract information about the file

```
ffmpeg –s <size> -i <filename>
```

   - Video Codec
   - pixel format
   - size
   - PAR, Pixel Aspect Ratio
   - DAR, Display Aspect Ratio
   - fps
   - tbn = the time base in AVStream that has come from the container
   - tbc = the time base in AVCodecContext for the codec used for a particular stream
   - tbr = tbr is guessed from the video stream and is the value users want to see when they look for the video frame rate

2. Notice the size of the file:

```
ls –sh <filename>
```

```
ls –sh
```

Notice it's the result of:

$$(W \times H \times 8 \times 1.5 \times 300)/8$$

So, a frame for luminance, a quarter each for the two crominance, 8 bpp, 300 frames (duration of the video), divided by 8 (bit → byte).

3. Compress the video, for example mpeg2

```
ffmpeg –s <size> -i <filename.yuv> -b <desired bitrate> -y <filename.mpg>
```

Default settings, so:

- 25 fps
- yuv420 pixel format
- mpeg1video encoder
- 1 - 50 - 200 - 1000 kb/s bitrate for video (NB, if the video doesn't need the expressed bitrate, it'll use less)

4. Check the difference in size between the original (YUV one) and the encoded one.

ls -la

PSNR

The phrase peak signal-to-noise ratio, often abbreviated PSNR, is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. Because many signals have a very wide dynamic range, PSNR is usually expressed in terms of the logarithmic decibel scale. The PSNR is most commonly used as a measure of quality of reconstruction of lossy compression codecs (e.g., for image compression). The signal in this case is the original data, and the noise is the error introduced by compression.

It is most easily defined via the mean squared error (**MSE**) which for two *m*×*n* monochrome images *I* and *K* where one of the images is considered a noisy approximation of the other is defined as:

$$MSE = \frac{1}{m\,n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2$$

The PSNR is defined as:

$$\begin{aligned} PSNR &= 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right) \\ &= 20 \cdot \log_{10} \left(\frac{MAX_I}{\sqrt{MSE}} \right) \end{aligned}$$

This time add two more options to the encoding string

- -psnr, shows the psnr value of the compressed frame
- -vstats, dump the video coding statistics to file

As you can see, according to the bitrate we provide to the encoder, the value of the psnr will fluctuate from 25 to 43.

GOP

Now let's see the difference in encoding in different way, playing with the insertion of I,P and B frames:

- **Iframes** are the least compressible but don't require other video frames to decode.
- **Pframes** can use data from previous frames to decompress and are more compressible than Iframes.
- **Bframes** can use both previous and forward frames for data reference to get the highest amount of data compression.

First example, only I-Frames at Highest quality

```
ffmpeg -s <size> -i <filename.yuv> -qscale 1 -qmin 1 -intra  <outputfile.mpg>
```

- -qscale, Use fixed video quantizer scale (VBR), maximum 1.
- -qmin, minimum 1.
- -intra, Use only intra frames.

Second example, only I- and P-Frames

```
ffmpeg -s <size> -i <filename.yuv> -g 25  <outputfile.mpg>
```

- Choose the size of the GOP

Third example, all together

```
ffmpeg -s <size> -i <filename.yuv> -g 25 -bf  4  <outputfile.mpg>
```

- -bf 4, Use 4 b-frames between Pframes

H264

H.264/MPEG-4 Part 10 or AVC (Advanced Video Coding) is a standard for video compression, and is currently one of the most commonly used formats for the recording, compression, and distribution of high definition video. The final drafting work on the first version of the standard was completed in May 2003.

H.264/MPEG-4 AVC is a block-oriented motion-compensation-based codec standard developed by the ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC Moving Picture Experts Group (MPEG). It was the product of a partnership effort known as the Joint Video Team (JVT). The ITU-T H.264 standard and the ISO/IEC MPEG-4 AVC standard (formally, ISO/IEC 14496-10 - MPEG-4 Part 10, Advanced Video Coding) are jointly maintained so that they have identical technical content.

H.264 is perhaps best known as being one of the codec standards for Blu-ray Discs; all Blu-ray players must be able to decode H.264. It is also widely used by streaming internet sources, such as videos from Vimeo, YouTube, and the iTunes Store, web software such as the Adobe Flash Player and Microsoft Silverlight, broadcast services for DVB and SBTVD, direct-broadcast satellite television services, cable television services, and real-time videoconferencing.

The H.264 name follows the ITU-T naming convention, where the standard is a member of the H.26x line of VCEG video coding standards; the MPEG-4 AVC name relates to the naming convention in ISO/IEC MPEG, where the standard is part 10 of ISO/IEC 14496, which is the suite of standards known as MPEG-4. The standard was developed jointly in a partnership of VCEG and MPEG, after earlier development work in the ITU-T as a VCEG project called H.26L. It is thus common to refer to the standard with names such as H.264/AVC, AVC/H.264, H.264/MPEG-4 AVC, or MPEG-4/H.264 AVC, to emphasize the common heritage. Occasionally, it is also referred to as "the JVT codec", in reference to the Joint Video Team (JVT) organization that developed it. (Such partnership and multiple naming is not uncommon. For example, the video codec standard known as MPEG-2 also arose from the partnership between MPEG and the ITU-T, where MPEG-2 video is known to the ITU-T community as H.262.[1]) Some software programs (such as VLC media player) internally identify this standard as AVC1.

H.264/AVC/MPEG-4 Part 10 contains a number of new features that allow it to compress video much more effectively than older standards and to provide more flexibility for application to a wide variety of network environments. In particular, some such key features include:

- Multi-picture inter-picture prediction including the following features:
  - Using previously-encoded pictures as references in a much more flexible way than in past standards, allowing up to 16 reference frames (or 32 reference fields, in the case of interlaced encoding) to be used in some cases. This is in contrast to prior standards, where the limit was typically one; or, in the case of conventional "B pictures", two. This particular feature usually allows modest improvements in bit rate and quality in most scenes. But in certain types of scenes, such as those with repetitive motion or back-and-forth scene cuts or uncovered background areas, it allows a significant reduction in bit rate while maintaining clarity.
  - Variable block-size motion compensation (VBSMC) with block sizes as large as 16×16 and as small as 4×4, enabling precise segmentation of moving regions. The supported luma prediction block sizes include 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4, many of which can be used together in a single macroblock. Chroma prediction block sizes are correspondingly smaller according to the chroma subsampling in use.
  - The ability to use multiple motion vectors per macroblock (one or two per partition) with a maximum of 32 in the case of a B macroblock constructed of 16 4×4 partitions. The motion vectors for each 8×8 or larger partition region can point to different reference pictures.
  - The ability to use any macroblock type in B-frames, including I-macroblocks, resulting in much more efficient encoding when using B-frames. This feature was notably left out fromMPEG-4 ASP.

    - Six-tap filtering for derivation of half-pel luma sample predictions, for sharper subpixel motion-compensation. Quarter-pixel motion is derived by linear interpolation of the halfpel values, to save processing power.
    - Quarter-pixel precision for motion compensation, enabling precise description of the displacements of moving areas. For chroma the resolution is typically halved both vertically and horizontally (see 4:2:0) therefore the motion compensation of chroma uses one-eighth chroma pixel grid units.
    - Weighted prediction, allowing an encoder to specify the use of a scaling and offset when performing motion compensation, and providing a significant benefit in performance in special cases—such as fade-to-black, fade-in, and cross-fade transitions. This includes implicit weighted prediction for B-frames, and explicit weighted prediction for P-frames.
- Spatial prediction from the edges of neighboring blocks for "intra" coding, rather than the "DC"-only prediction found in MPEG-2 Part 2 and the transform coefficient prediction found inH.263v2 and MPEG-4 Part 2. This includes luma prediction block sizes of 16×16, 8×8, and 4×4 (of which only one type can be used within each macroblock).
- Lossless macroblock coding features including:
    - A lossless "PCM macroblock" representation mode in which video data samples are represented directly,[15] allowing perfect representation of specific regions and allowing a strict limit to be placed on the quantity of coded data for each macroblock.
    - An enhanced lossless macroblock representation mode allowing perfect representation of specific regions while ordinarily using substantially fewer bits than the PCM mode.
- Flexible interlaced-scan video coding features, including:
    - Macroblock-adaptive frame-field (MBAFF) coding, using a macroblock pair structure for pictures coded as frames, allowing 16×16 macroblocks in field mode (compared with MPEG-2, where field mode processing in a picture that is coded as a frame results in the processing of 16×8 half-macroblocks).
    - Picture-adaptive frame-field coding (PAFF or PicAFF) allowing a freely-selected mixture of pictures coded either as complete frames where both fields are combined together for encoding or as individual single fields.
- New transform design features, including:
    - An exact-match integer 4×4 spatial block transform, allowing precise placement of residual signals with little of the "ringing" often found with prior codec designs. This is conceptually similar to the well-known DCT design, but simplified and made to provide exactly-specified decoding.
    - An exact-match integer 8×8 spatial block transform, allowing highly correlated regions to be compressed more efficiently than with the 4×4 transform. This is

conceptually similar to the well-known DCT design, but simplified and made to provide exactly-specified decoding.

    - Adaptive encoder selection between the 4×4 and 8×8 transform block sizes for the integer transform operation.
    - A secondary Hadamard transform performed on "DC" coefficients of the primary spatial transform applied to chroma DC coefficients (and also luma in one special case) to obtain even more compression in smooth regions.
- A quantization design including:
    - Logarithmic step size control for easier bit rate management by encoders and simplified inverse-quantization scaling
    - Frequency-customized quantization scaling matrices selected by the encoder for perceptual-based quantization optimization
- An in-loop deblocking filter that helps prevent the blocking artifacts common to other DCT-based image compression techniques, resulting in better visual appearance and compression efficiency
- An entropy coding design including:
    - Context-adaptive binary arithmetic coding (CABAC), an algorithm to losslessly compress syntax elements in the video stream knowing the probabilities of syntax elements in a given context. CABAC compresses data more efficiently than CAVLC but requires considerably more processing to decode.
    - Context-adaptive variable-length coding (CAVLC), which is a lower-complexity alternative to CABAC for the coding of quantized transform coefficient values. Although lower complexity than CABAC, CAVLC is more elaborate and more efficient than the methods typically used to code coefficients in other prior designs.
    - A common simple and highly structured variable length coding (VLC) technique for many of the syntax elements not coded by CABAC or CAVLC, referred to as Exponential-Golomb coding (or Exp-Golomb).
- Loss resilience features including:
    - A Network Abstraction Layer (NAL) definition allowing the same video syntax to be used in many network environments. One very fundamental design concept of H.264 is to generate self contained packets, to remove the header duplication as in MPEG-4's Header Extension Code (HEC).[16] This was achieved by decoupling information relevant to more than one slice from the media stream. The combination of the higher-level parameters is called a parameter set.[16] The H.264 specification includes two types of parameter sets: Sequence Parameter Set (SPS) and Picture Parameter Set (PPS). An active sequence parameter set remains unchanged throughout a coded video sequence, and an active picture parameter set remains unchanged within a coded picture. The sequence and picture parameter set structures contain information such as

picture size, optional coding modes employed, and macroblock to slice group map.[16]

    - Flexible macroblock ordering (FMO), also known as slice groups, and arbitrary slice ordering (ASO), which are techniques for restructuring the ordering of the representation of the fundamental regions (*macroblocks*) in pictures. Typically considered an error/loss robustness feature, FMO and ASO can also be used for other purposes.
    - Data partitioning (DP), a feature providing the ability to separate more important and less important syntax elements into different packets of data, enabling the application of unequal error protection (UEP) and other types of improvement of error/loss robustness.
    - Redundant slices (RS), an error/loss robustness feature allowing an encoder to send an extra representation of a picture region (typically at lower fidelity) that can be used if the primary representation is corrupted or lost.
    - Frame numbering, a feature that allows the creation of "sub-sequences", enabling temporal scalability by optional inclusion of extra pictures between other pictures, and the detection and concealment of losses of entire pictures, which can occur due to network packet losses or channel errors.
- Switching slices, called SP and SI slices, allowing an encoder to direct a decoder to jump into an ongoing video stream for such purposes as video streaming bit rate switching and "trick mode" operation. When a decoder jumps into the middle of a video stream using the SP/SI feature, it can get an exact match to the decoded pictures at that location in the video stream despite using different pictures, or no pictures at all, as references prior to the switch.
- A simple automatic process for preventing the accidental emulation of start codes, which are special sequences of bits in the coded data that allow random access into the bitstream and recovery of byte alignment in systems that can lose byte synchronization.
- Supplemental enhancement information (SEI) and video usability information (VUI), which are extra information that can be inserted into the bitstream to enhance the use of the video for a wide variety of purposes.[*clarification needed*]
- Auxiliary pictures, which can be used for such purposes as alpha compositing.
- Support of monochrome, 4:2:0, 4:2:2, and 4:4:4 chroma subsampling (depending on the selected profile).
- Support of sample bit depth precision ranging from 8 to 14 bits per sample (depending on the selected profile).
- The ability to encode individual color planes as distinct pictures with their own slice structures, macroblock modes, motion vectors, etc., allowing encoders to be designed with a simple parallelization structure (supported only in the three 4:4:4-capable profiles).

- Picture order count, a feature that serves to keep the ordering of the pictures and the values of samples in the decoded pictures isolated from timing information, allowing timing information to be carried and controlled/changed separately by a system without affecting decoded picture content.

These techniques, along with several others, help H.264 to perform significantly better than any prior standard under a wide variety of circumstances in a wide variety of application environments. H.264 can often perform radically better than MPEG-2 video—typically obtaining the same quality at half of the bit rate or less, especially on high bit rate and high resolution situations.[17]

Like other ISO/IEC MPEG video standards, H.264/AVC has a reference software implementation that can be freely downloaded.[18] Its main purpose is to give examples of H.264/AVC features, rather than being a useful application *per se*. Some reference hardware design work is also under way in the Moving Picture Experts Group. The above mentioned are complete features of H.264/AVC covering all profiles of H.264. A profile for a codec is a set of features of that codec identified to meet a certain set of specifications of intended applications. This means that many of the features listed are not supported in some profiles. Various profiles of H.264/AVC are discussed in next section.

Profiles

The standard defines 17 sets of capabilities, which are referred to as profiles, targeting specific classes of applications. If you want to know exactly what they are used for, you can find all the 17 online, just on Wikipedia, here we just mention that we can have

Constrained Baseline Profile (CBP)

Primarily for low-cost applications, this profile is most typically used in videoconferencing and mobile applications. It corresponds to the subset of features that are in common between the Baseline, Main, and High Profiles described below.

Baseline Profile (BP)

Primarily for low-cost applications that require additional data loss robustness, this profile is used in some videoconferencing and mobile applications. This profile includes all features that are supported in the Constrained Baseline Profile, plus three additional features that can be used for loss robustness (or for other purposes such as low-delay multi-point video stream compositing). The importance of this profile has faded somewhat since the definition of the Constrained Baseline Profile in 2009. All Constrained Baseline Profile bitstreams are also considered to be Baseline Profile bitstreams, as these two profiles share the same profile identifier code value.

Main Profile (MP)

This profile is used for standard-definition digital TV broadcasts that use the MPEG-4 format as defined in the DVB standard.[19] It is not, however, used for high-definition television broadcasts, as the importance of this profile faded when the High Profile was developed in 2004 for that application.

Extended Profile (XP)

Intended as the streaming video profile, this profile has relatively high compression capability and some extra tricks for robustness to data losses and server stream switching.

High Profile (HiP)

The primary profile for broadcast and disc storage applications, particularly for high-definition television applications (for example, this is the profile adopted by the Blu-ray Discstorage format and the DVB HDTV broadcast service).

and so many others…

Sample Encoder/Decoder

Retrieve the encoder/decoder at http://iphome.hhi.de/suehring/tml/ . You should download the latest package sw (mine was JM14**.zip) and the pdf of documenation, more or less 90 pgg.

Once you extracted it, follow the guide in the pdf so:

- chmod +u+x unixprep.sh
- make

then in Bin you'll have all the code. Remember to get build-essential package, if you never had to compile anything on a linux distro.

Once you have the code, the major elements in the Bin folder are:

- lencode.exe
- ldecode.exe
- encoder.cfg

Parameters to work on

This is a testing tool, so it encode not a proper h264 stream, but a sort of headerless file. So the right usage of this element is to compute in cascade first the encoder, then the decoder and evaluate the result, comparing the input and the decoded output. As you can see from the reference pdf file, there are a huge amount of parameters you can play with, when encoding a raw video into an h264 stream, but although it's an interesting thing to examine all of them, we will focus on an interesting subset of them.

Since, as I said the number of parameters is huge, an easy way to deal with them is to use a configuration file, like the one you can already find in the BIN folder. We will use the encoder.cfg . Obiously nothing stops you to practice on the other configuration files which you already find in the folder.

So in order to perform encoding/decoding, just type:

```
./lencode.exe  -f encoder.cfg
```

```
./decode.exe
```

You have to modify parameters into the configuration files before running it, like for example {you can see all the parameters explaination in the pdf that you can download from the site where you downloaded the encoder}:

- Input file, choosing the file to work on, like the files we used the last time: foreman sequence, temptete sequence, just take it raw, in order to better see the difference in encoding
- FramesToBeEncoded, since it's an expensive operation (the more you encode/ compress, the more time and computational power you use) it's better to limit the number of frame to process, depending on the power of your pc.
- SourceWidth/Height, to match the input
- Outputfiles names, if you want

The exercises in encoding we like you to do are the following.

PSlice Motion Estimation

Work on the parameters controlling the block sizes in motion compensation. Try to enable/ disable the elements [LINES 227-240]

Previous MPEG specifications allowed pictures to be coded as I-frames, P-frames, or B-frames. h.264 is more complex and wonderful. It allows individual frames to be coded as multiple slices, each of which can be of type I, P, or B, or even more esoteric types. As with previous MPEG specifications, in h.264 each slice consists of one or more 16×16 macroblocks. Each macroblock in our 4:2:0 sampling scheme contains 16×16 luma samples, and two 8×8 blocks of chroma samples. For this simple encoder, I won't be compressing the video data at all, so the samples will be directly copied into the h.264 output.

Try to enable/disable the P-SLICE parameters, 0/1 elements, to see the results. Remember to put constraints on the number of frames to be encoded, in order to contain the time used to encode.

Number of reference frames

It's the number of reference frames stored in the Decoded Picture Buffer (DPB) for motion estimation and compensation. [LINE 167]

Using previously-encoded pictures as references in a much more flexible way than in past standards, allowing up to 16 reference frames (or 32 reference fields, in the case of interlaced encoding) to be used in some cases. This is in contrast to prior standards, where the limit was typically one; or, in the case of conventional "B pictures", two. This particular feature usually allows modest improvements in bit rate and quality in most scenes. But in

certain types of scenes, such as those with repetitive motion or back-and-forth scene cuts or uncovered background areas, it allows a significant reduction in bit rate while maintaining clarity.

This allows the video encoder to choose among more than one previously decoded frame on which to base each macroblock in the next frame. While the best frame for this purpose is usually the previous frame, the extra reference frames can improve compression efficiency and/or video quality. Note that different reference frames can be chosen for different macroblocks in the same frame. The maximum number of concurrent reference frames supported by H.264 is 16.

Try to modify the parameters and see the difference.

GOP width

Try to modify the value of GOP

PrimaryGopLength [LINE 347]: 1—16

With longer GOP I can encode and compress more, but I need to have more frame for real time application. TradeOff storage/RealTime.

Rate Control

Rate Control enable/disable [Line 441]

Different algorithm for rate control [RCUpdateMode 0-1-2-3]

A rate control algorithm dynamically adjusts encoder parameters to achieve a target bitrate. It allocates a budget of bits to each group of pictures, individual picture and/or sub-picture in a video sequence. Rate control is not a part of the H.264 standard, but the standards group has issued non-normative guidance to aid in implementation. The purpose of this white paper is to offer 1) a basic understanding of what rate control is and why it is essential and 2) a common framework and terminology so that schemes originating from H.264 and other standards groups can be more easily understood and compared.

DeBlock filter control

A deblocking filter is applied to blocks in decoded video to improve visual quality and prediction performance by smoothing the sharp edges which can form between macroblocks when block coding techniques are used. The filter aims to improve the appearance of decoded pictures. The filter operates on the edges of each 4×4 or 8×8 transform block in the luma and chroma planes of each picture. Each small block's edge is assigned a boundary strength based on whether it is also a macroblock boundary, the coding (intra/inter) of the blocks, whether references (in motion prediction and reference frame choice) differ, and

whether it is a luma or chroma edge. Stronger levels of filtering are assigned by this scheme where there is likely to be more distortion. The filter can modify as many as three samples on either side of a given block edge (in the case where an edge is a luma edge that lies between different macroblocks and at least one of them is intra coded). In most cases it can modify one or two samples on either side of the edge (depending on the quantization step size, the tuning of the filter strength by the encoder, the result of an edge detection test, and other factors).

Try to play with the parameters in I – P – B frames,enabling the filter and modify the parameters, and notice the differences.

Error Resilience

Loss resilience features including:

- A Network Abstraction Layer (NAL) definition allowing the same video syntax to be used in many network environments. One very fundamental design concept of H.264 is to generate self contained packets, to remove the header duplication as in MPEG-4's Header Extension Code (HEC).[16] This was achieved by decoupling information relevant to more than one slice from the media stream. The combination of the higher-level parameters is called a parameter set.[16] The H.264 specification includes two types of parameter sets: Sequence Parameter Set (SPS) and Picture Parameter Set (PPS). An active sequence parameter set remains unchanged throughout a coded video sequence, and an active picture parameter set remains unchanged within a coded picture. The sequence and picture parameter set structures contain information such as picture size, optional coding modes employed, and macroblock to slice group map.[16]
- Flexible macroblock ordering (FMO), also known as slice groups, and arbitrary slice ordering (ASO), which are techniques for restructuring the ordering of the representation of the fundamental regions (*macroblocks*) in pictures. Typically considered an error/loss robustness feature, FMO and ASO can also be used for other purposes.
- Data partitioning (DP), a feature providing the ability to separate more important and less important syntax elements into different packets of data, enabling the application of unequal error protection (UEP) and other types of improvement of error/loss robustness.
- Redundant slices (RS), an error/loss robustness feature allowing an encoder to send an extra representation of a picture region (typically at lower fidelity) that can be used if the primary representation is corrupted or lost.

- Frame numbering, a feature that allows the creation of "sub-sequences", enabling temporal scalability by optional inclusion of extra pictures between other pictures, and the detection and concealment of losses of entire pictures, which can occur due to network packet losses or channel errors.