Image and Video Compression HW 2

1 Reading and Writing Video Files

The *yuv* format serves to store raw video, much like a sequence of bit maps. It has no header, thus it's not readable without knowing parameters such as the frame size. The *y4m* format is yuv in a container that includes a video header and a short frame header, and is playable by some video players, for example vlc player. We will use these formats as input and output, thus we need to verify that we can read, write and display.

For your convenience, find three m files as follows.

1. yuv_read_frame: accepts a file name, frame index, and the frame parameters (width, height, chroma sampling, bit depth). Assumes yuv or y4m according to the file extension. Goes to the right place in the file and reads the frame into a structure that has y, cr and cb components: first is grayscale, others give color.

2. write_yuv_frame: accepts a single frame (y, cr, cb) and appends it to a file.

3. yuv_to_y4m: a script which uses the above both. It writes a yuv header, then reads frames from a y4m file (can also be yuv) one by one according to some range, then appends each frame to the output y4m file.

Do the following steps:

1. Go to http://ultravideo.fi/#testsequences. Choose and download a video you like, as long as it's available in 1080p resolution (4k will give huge files), bit depth 8 and format yuv-raw.

2. Apply yuv_to_y4m and verify that you get a video of length 100 frames that you can play using vlc player.

3. Just to make sure that you master the format, change the function to get fast-forward (write to the file every 5th frame) and rewind (frames in reverse order).

4. For the sequel we need to make the video grayscale. To that end, between reading and writing set the cr and cb parts to be all 128 (middle of 8-bit range, representing zero). Watch it in vlc, to verify that indeed you get a B&W clip.

2 Zero-motion coding

From here on you can use a video that's shorter than 100 frames (to save on running time). In this part we compare two approaches: in the first we encode each frame separately, using the encoder of Ex. 2, and in the second we encode the difference between the new frame and the compressed version of the previous one.

2.1 Separate Encoding

Read the frames from the y4m file (grayscale). For each frame, use the encoder of Ex. 2. Set the quantization step size (fixed for all the video) such that the PSNR is around 32dB. Write the reconstructed (grayscale, after DCT, quantization, IDCT etc) frames to a yuv file, and display it. What is the compressed rate in bits/pixel?

2.2 Encoding of Residual

Compress the first frame as above. Then loop over the frames, performing the following:

1. reference frame is the reconstructed version of the previous one.
2. read new frame
3. residual frame is the new frame minus (per pixel) the reference frame
4. compress residual frame using the encoder of Ex. 2, without DC prediction
5. reference frame is the old reference plus the compressed residual
6. write reference frame to file and repeat

With the same quantization step qs above, what is the compression rate? Compare.