Assignment 2

Course: TTT4135 - Multimedia signal processing

Authors: Group 10 - Sakaria A., Gard H. B. & Sander A.

Version: 1.0 | Date: February 28, 2022

Contents

1 Part 1 - Theoretical

1.1 Task 1 - Temporal Prediction

1.1.a

Motion compensation utilizes the temporal redundancy in a stream of video pictures to predict which part of the image that will stay the same, and which part that will most likely move. In other words it retrieves the residual between the past and current frame to compress the information in the video. This is done by looking at one or a multiple of past or future frames to determine which part of the image that will change by finding the difference, then finding the residual.

1.1.b

I, P and B frames are part of the GOP in a video stream. The I-frames or Intra-frames contains the most information and are the key frames in the stream. The frames that undergo motion compensation is often compared to the I-frames in the start of a video stream. As there is only performed coding for spatial redundancy (usually by a DCT-transform on the frame) on the I-frames, the compression is minimal on these frames.

P-frames or predicted frames are what we predict will happen on the next frame based on the previous frame from doing motion estimation and compensation. Motion compensation is a key component in making the P-frame, as the sum of the motion compensation and the input image/frame results in the predicted frame. These frames are both compressed in the spatial (DCT) and temporal domain, as the output of the P-frame is only the change that occurred from the last frame(s) up until the current frame. Hence these frames are quite heavily compressed.

The B-frame is somewhat similar to the P-frame, although the B-frame (bi-directional frame) can utilize motion estimation both forward or backwards in time. These frames contain an estimate of the movement that would most likely happen between the past and future frame. Hence these frames are also heavily compressed.

1.1.c

The *transmission order* is ordered in the way the images needs to be processed. Hence we usually get an I-frame followed by a P-frame, than a B-frame etc. This is done because the B-frames needs both the forward and backward frame to be evaluated before the B-frame can be evaluated. The *display order* is organized in the way the frames will be shown on the screen.

1.1.d

H.264 uses motion estimation with a varying block size around important features in the frame. Which helps to avoid blocking/striping/ringing/tearing artifacts to be produced around important features in the B- and P-frames. These artifacts can often easily be picked up by the human eye and will reduce the quality of the video stream for the viewer and in general.

1.2 Task 2 - Decomposition/Transform

1.2.a

Classic video coders have usually used the Discrete Cosine Transform (DCT) to exploit spatial redundancy. Often with 8x8 blocks. H.264 uses a similar method, but with 4x4 blocks and a separable integer transform with similar properties as DCT.

1.2.b

For minimization of blocking artifacts in the video stream, H.264 has a blocking filter implemented in its codec. And as mentioned above H.264 uses a smaller block size for the DCT, this results in that the details in the image is less averaged out, than what it would have been with a block size of 8x8 pixels.

2 Part 2 - Practical

2.1 Task 3 - Video coding

Since the quantized error are entropy coded the distortion rate D_R is given by the following equation where R is the rate and σ_e is the prediction error.

$$D(R) = \frac{\pi e}{6}\sigma_e 2^{-2R} \tag{1}$$

We then need to find prediction error first to find the distortion rate. We can find the prediction error by minimizing the mean squared error estimator.

$$\sigma_e = E[(Z[N] - Z'[n])^2] \tag{2}$$

We can calculate this prediction error for both predictors and add the prediction errors together to get the total prediction error and calculate the total distortion rate for the coding method.

Calculating the prediction error for the first predictor σ_{e1}.

$$
\begin{aligned}
\sigma_{e1} &= E[(Z[N] - Z'[n])^2] \\
&= E[Z^2[N] - Z[N]Z'[N] + Z'^2[N]] \\
&= E[Z^2[N]] - E[Z[N]Z'[N]] + E[Z'[N]^2] \\
&= r_Z(0) - E[Z[N] \cdot a \cdot Z[N+1]] + E[a^2 Z^2[N]] \\
&= r_Z(0) - a r_Z(1) + a^2 r_z(0) \\
&= 1 - 0.9a + a^2
\end{aligned}
$$

We can then minimize the equation we got above and get the prediction error.

$$
\begin{aligned}
\partial/\partial a &= 2 \cdot a - 0.9 \\
\longrightarrow a &= 0.45 \\
\longrightarrow \sigma_e &= 1 - 0.9 \cdot 0.45 + 0.45^2 \\
\longrightarrow \sigma_e &= \frac{319}{400} = 0.7975
\end{aligned}
$$

Now we do the same for the other predictor

$$
\begin{aligned}
\sigma_{e2} &= E[(Z[N] - Z'[n])^2] \\
&= E[Z^2[N] - Z[N]Z'[N] + Z'^2[N]] \\
&= E[Z^2[N]] - E[Z[N]Z'[N]] + E[Z'[N]^2] \\
&= r_Z(0) - E[Z[N] \cdot (b \cdot Z[N-1] + c \cdot Z[N+1]) \\
&+ E[b^2 Z^2[N-1]] + E[C^2 Z^2[N+1]] + E[b \cdot c \cdot Z^2[N+1]Z[N-1]] \\
&= r_Z(0) - b r_Z(1) - c r_Z(0) + b^2 r_z(0) + c^2 r_z(0) + bc \cdot r_Z[2] \\
&= c^2 + b^2 - 0.9b - 0.9c + 0.81bc + 1
\end{aligned}
$$

And minimize this expression too. We also know that $b = c$ since the expression is symmetrical.

$$
\begin{aligned}
d/db &= 2b - 0.9 + 0.81c \\
\longrightarrow b &= 0.45 - 0.405c \\
d/dc &= 2c - 0.9 + 0.81b \\
\longrightarrow c &= 0.45 - 0.405b \\
\longrightarrow b &= 0.45 - 0405(0.45 - 0.405b) \\
\longrightarrow b &= \frac{90}{281} \\
\longrightarrow c &= \frac{90}{281} \\
\longrightarrow \sigma_e &= \frac{200}{281} = 0.7117
\end{aligned}
$$

Finally we can use the equation for the predictor to get the distortion rate:

$$
\begin{aligned}
D(R) &= \frac{\pi e}{6}\sigma_e 2^{-2R} \\
D(R) &= \frac{\pi e}{6}(\sigma_{e1} + \sigma_{e2})2^{-2R} \\
D(R) &= 0.2515\pi e 2^{-2R}
\end{aligned}
$$

2.2 Task 4 - Motion estimation

See appendix (section 3) for link to python code.

Given the following pictures, plot the motion estimation vectors for the movement between the images.

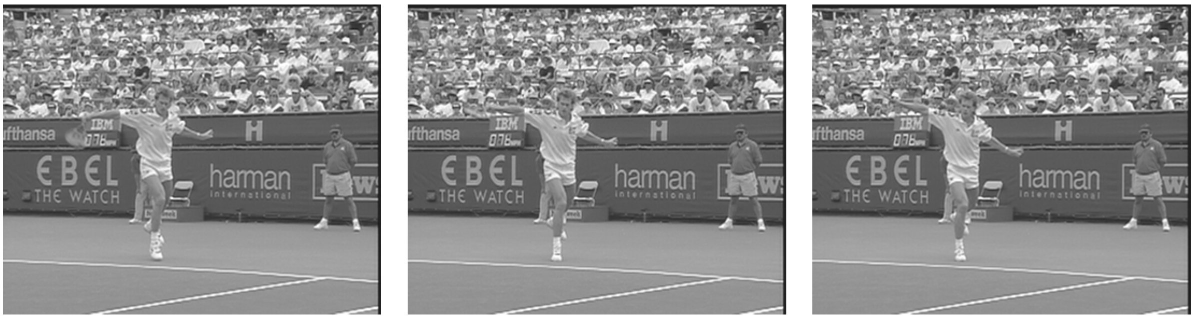

Figure 1: The given images.

2.2.a

Below in figure 2 we have the motion estimation vectors resulting from the implementation of the optimal displacement function. For these plots we used macro blocks of size 16x16 px, and a search range off 32x32 px. Here we see some signs of movement around where the tennis player is moving, whereas the arrows point at where the block was moved from.

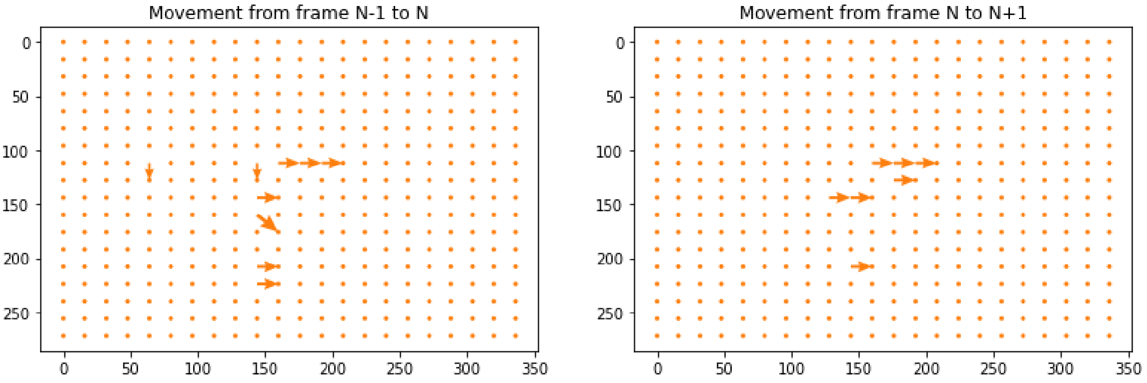

Figure 2: Motion estimation for the 16x16 macro blocks.

2.2.b

Below in figure 3 we see the results from using 8x8 px blocks instead of using the 16x16 px macro blocks. Compared with the plots in figure 2, we can see more of the fine movements by the tennis player. The main reason for the increased detail is the wider search area per block, as the 16x16 blocks are compared with the block to the right, the one below, and the one that is both to the right and down. For the 8x8 block the search is done in 17 operations, while the 16x16 only uses 3 operations for comparing with the origin macro block.

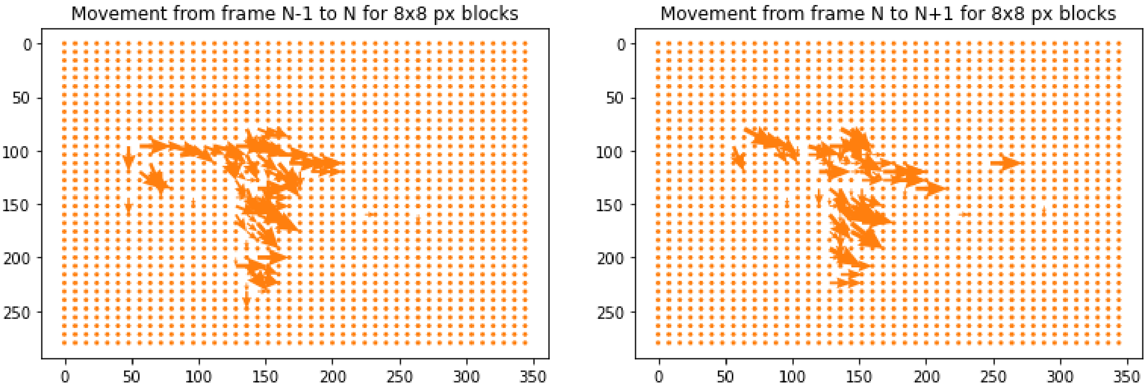

Figure 3: Motion estimation for the 8x8 blocks. It might be a bit hard to read, but couldent really figure out how to make the arrows have the same size, without loosing most of them.

3 Appendix

Link to python code:
https://github.com/saa96/TTT4135_MMS/blob/master/O2/Assignment2.py