EQ2330 Image and Video Processing - Project 3

Stefano Imposcopi
imoscopi@kth.se

Denis Kulicek
kulicek@kth.se

January 8th, 2016

1) INTRODUCTION

The goal of this project is to evaluate the performance of three different video encoding schemes.

1. **Intra-frame video coder**: exploiting only the spatial redundancy of the single frame to reduce the bandwidth.
2. **Conditional replenishment video coder**: adding the possibility of copying from the past frames to exploit the correlation between successive frames and save extra bandwidth without sacrificing quality.
3. **Inter-frame video coder with motion compensation:** extension of the 2nd coder, using motion compensation with shifts of integer pixels to be able to remove the blocking artifacts given by shifts between successive blocks.

The structure and performance of each coder are described and compared with the other coders. PSNR (with MSE distortion) is used as the performance measure.

The lower bound given by the Shannon Entropy is used to estimate the bit rates of 3 variable length encoders used for the DCT coefficients of the image, the DCT coefficients of the motion compensated residuals and the motion vectors. This estimated and computed measures are finally used to investigate the trade-off between performance and bandwidth consumption typical of every coding scheme and compare it with theoretical results.

Reminder, without any compression, using 8bits per pixel would result in a video with a bitrate of roughly 6080 Kbps. Video size used in this project is 176 pixels x 144 pixels, with a frame rate of 30 frames per second.

2) INTRA-FRAME VIDEO CODER

This mode encodes each frame independently from previous frames, in other words, it considers video purely as a set of uncorrelated images and does not utilize correlation within subsequent frames for compression. Even in this simple case - the bitrate reduction is significant, for quantization step of 8 bitrate is reduced to less than 1100 Kbps.

Intra frame compression plays a very important part, necessary in every video encoder since the spatial redundancy must be exploited to reduce the bitrate. The frame is divided into smaller blocks of 16x16 pixels, on which four 8x8 DCTs are applied. The block DCT works pretty well for quantization step up to 16, but after the blocking distortion tends to become quite noticeable, as shown in the second image of Figure 2.1.

A transform on the whole frame like a DWT could be preferred to tackle these small 8x8 blocking artifacts.

Figure 2.1 - 8x8 DCT compressed images, with Q step of 8 and 32 respectively

Instead of relying purely on spatial redundancy to code the video, we can exploit the fact that in most cases subsequent frames in the video are usually correlated, and therefore there are many smart techniques, which can be applied to achieve further compression, without sacrificing the PSNR.

Next technique shows one way to achieve this by marking blocks of the frame that can be reused, instead of transmitting the block information again.

3) CONDITIONAL REPLENISHMENT VIDEO CODER

An extra encoding mode is introduced. This time the already inter coded frames are analyzed in blocks of 16x16 and the encoder has the possibility of transmitting the frame normally (as in the first encoder) or signaling that the block for the next frame can be copied from the same block in the previous frame. This makes sense since **successive frames are in most cases strongly correlated** and usually there are **many blocks that are almost identical**. At each block **1 extra bit is necessary to signal the choice** between intra mode and copy mode.

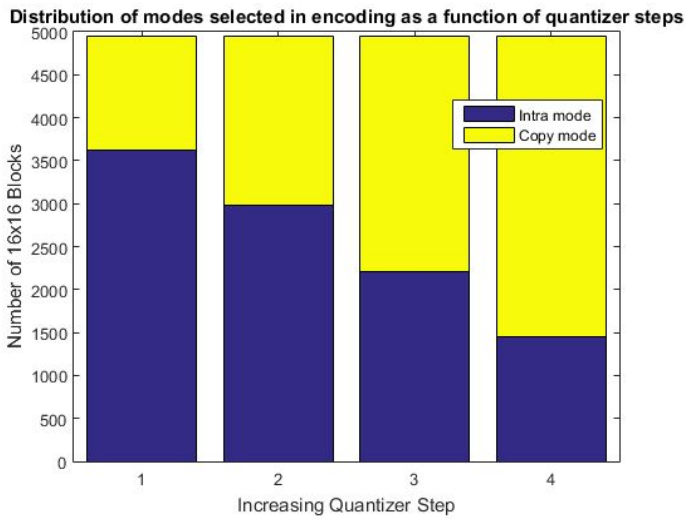

Figure 3.1 - Stacked bar chart showing mode selection for increasing quantizer steps (8,16,32,64)

The figure 3.1 below shows the amount of 16x16 blocks for the “Foreman” video that can be copied over from the previous frame instead of being replenished. If we take for instance quantizer step

size 8 (which is represented by the leftmost bar in the chart below), we can see that the amount of copied blocks is roughly 30%. Having this in mind, if we then compare bitrates of encoder 1 *(only intra mode)* and encoder 2 *(intra + copy mode)* for the same quantizer size - we can see that the savings in bitrate matches the rough 30%, however it comes at the cost of PSNR decrease of roughly 2dB - this can be seen in Figure 4.1.

To make a choice at each block, the encoder chooses the mode minimizing the **Lagrangian cost**

$$J_n = D_n + \lambda R_n \quad n = 1, 2.$$

Where D_n is the MSE distortion and R_n is the rate in bits/block to encode with mode n.
$\lambda = k \cdot Q^2$ is the Lagrange multiplier that balances the tradeoff between MSE distortion and rate. The lagrange multiplier is proportional to the square of the quantization step, and we manually tuned the k value to achieve an approximately equal contribution of distortion and rate to the cost function.

With this parameter set, we can see in Figure 3.1 that both modes are selected in encoding the video “Foreman”. When the quantization gets coarser, at Q = 64, we get that the copy mode is largely preferred. This is probably because the block distortion introduced by the DCT is really big, there is not much detail anymore in the video, and thus the loss of quality due to the copy mode is negligible compared to the huge saving in rate, since the copy mode is the one that saves most bandwidth, with R2 = 1 bit/block.

4) INTEGER-PEL MOTION COMPENSATION VIDEO CODER

In the last part we extended the previous encoder with a third mode (called *inter mode*) based on integer-pel motion compensation for the prediction of the frame. 2 extra bits/block are thus needed to signal a choice between 3 modes.

When the new inter mode is selected, the decoder receives a motion vector for the 16x16 block, and a quantized residual block. It uses the motion vectors to take the best block possible from the previous frame, and it adds to it the quantized residual to try to match the current block as close as possible. Mathematically,

$$f(x,y,z+1) = f(x+dx, y+dy, z) + r(x,y,z+1) \quad for\ (x,y) \in Block(16x16)$$

Where (dx, dy) is the motion vector chosen from all 442 possible shifts given by (-10,10)x(-10x10). The rationale for this mode is that by looking in the 10 pixel neighbourhood of the block to compute, we are likely to find a very similar block in the previous frame, even when an object has had a shift of up to 10 pixels between the two frames.

The residual $r(x,y)$ is the difference between the original unaltered block and the predicted motion compensated block and is necessary to get rid of strong (16x16) blocking artifacts that can arise, since each block can have a completely different motion vector and the image reconstructed using overlapping blocks from the previous frame. The residual ideally gets rid of the distortion of an imperfect prediction.
In our encoded the residual is quantized using the same block DCT scheme used for the intra mode in part 2.

Another key insight is that the residual for this mode is much closer to a zero image than when using the copy mode. This is because the **motion vector is chosen such to minimize the MSE between real block and predicted block**, which is equivalent to minimizing the energy of the residual across all possible shifts.

This is clearly visible in Figure 4.1., where gray corresponds to values close to 0. We can see that the second residual, with motion compensation, is mostly composed of zero values, except at the most important edges which have changed due to movements between frames that are too difficult to match with such a simple block-translation model. The inter mode residual has so a smaller variance and entropy compared to the first residual obtained from copying directly from the previous frame.

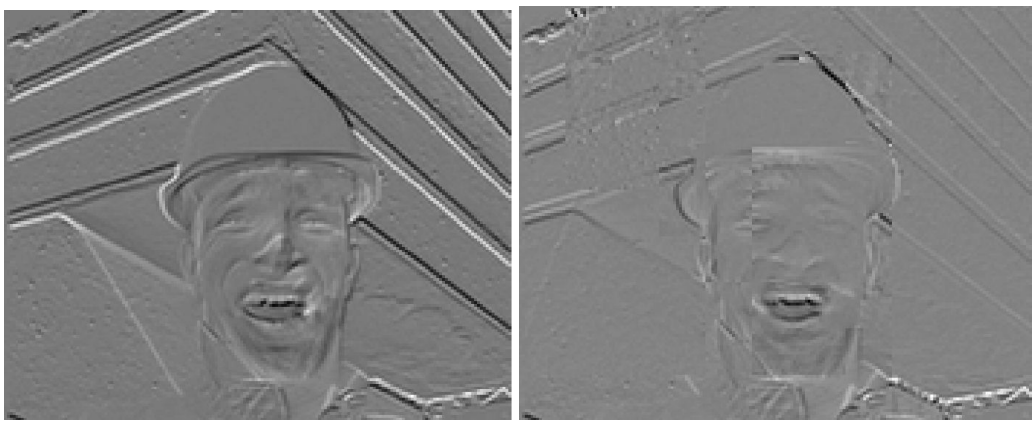

Figure 4.1 - Residual Image without and with Motion Compensation Respectively

This **low entropy of the motion compensated residual** allows for a quantization with a small bitrate. Using the entropy as the lower bound of the rate, we get a rate for the residual which is **always less than 1 bit/pixel**, even for a small quantization step of 8.

The same is true for the motion vectors, which are mostly distributed around the zero shift, given that most blocks do not change completely from one frame to the other (see Figure 4.2)

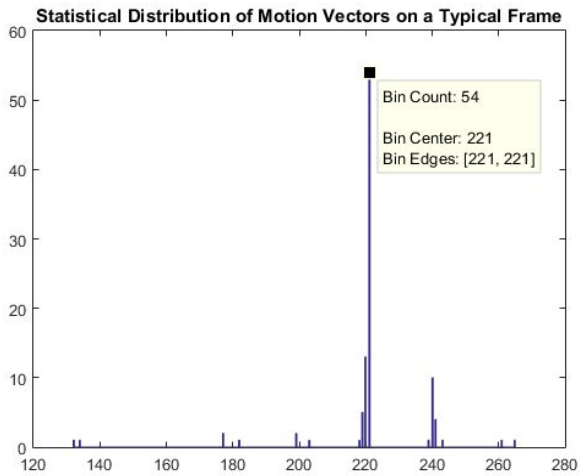

Figure 4.2 - Histogram of Motion Vectors. The index 221 corresponds to shift (dx,dy) = (0,0)

On a typical frame, half or more motion vectors are (0,0). This allows for a good bandwidth reduction using entropy coding. From 8 bits/vector needed for a fixed length encoder we get approximately 1 bit/vector using a variable length coder like Huffman.

This allows the inter mode to obtain a good quality in reconstructing the image without excessive “shift-blocking”, while at the same time using a reasonable amount of bandwidth, lower than the one needed for the intra mode (transmitting the actual block).

For example for a Quantization step of 16, we estimate 117 bits/block for the inter mode, significatively lower than the 316 bits/block needed for the intra mode.

For each block, the encoder chooses the best mode through the minimization of the Lagrangian cost function, as already explained in part 2.

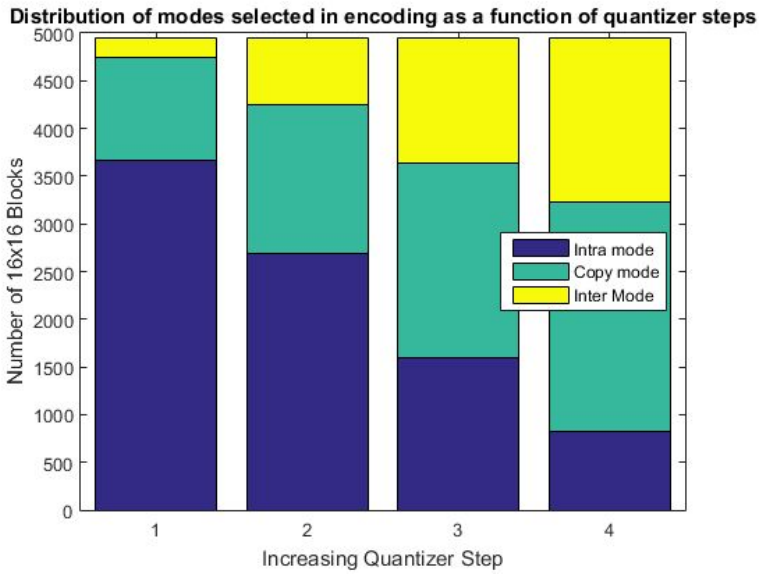

Figure 4.3 - Stacked bar chart showing mode selection for increasing quantizer steps (8,16,32,64)

As seen in Figure 4.3, all modes are selected for each quantization step. As for encoder 2 (Figure 3.1), the more quantization distortion is introduced by a larger Q step, the more the reduction in bandwidth tends to be favoured, and thus the intra mode is rarely used (only on the very complex blocks that cannot be predicted at all, not even with motion compensation).

Already with Q=32 most blocks are predicted from the previous frames. The most simple ones are copied directly, with a huge saving in bandwidth (2bits/block only), while the ones that cannot be copied due to a shift between frames are predicted with motion compensation and rendered as similar to the original as possible by summing the residual signal.

COMPARISON OF CODERS AND CONCLUSIONS

We encoded the videos using 4 different quantization steps of DCT coefficients (8,16,32,64) and plotted the results in the PSNR-rate graph, to be able to investigate the performance of each video at different quality levels.

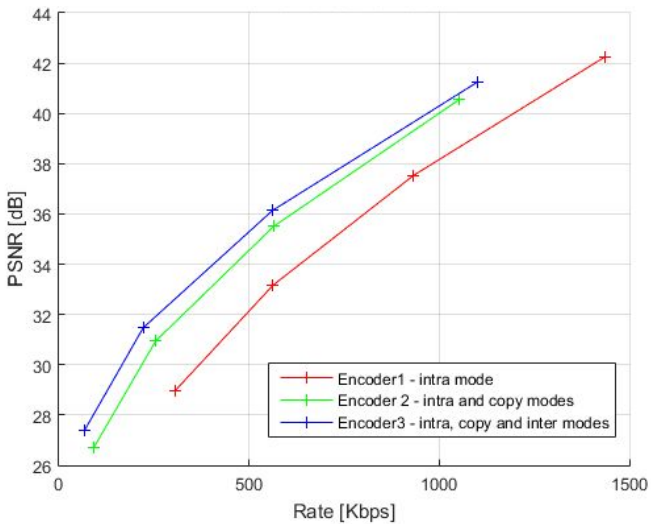

Figure 4.1. - Performance-bitrate plot for all encoders. Left are the measurements for Foreman video, right plot represents Mother & Daughter video

As expected for the intra mode only at high bitrates we have an increase of roughly 6dB/pixel, which makes sense since the noise used for computing the SNR is the MSE of a uniform quantizer. The results are thus in line with the rate/distortion theory.

The type 2 encoder obtains a reduction of up to 250 Kbps in bandwidth, at the same PSNR quality measure. It is a good gain since the copy mode is really simple in exploiting the time redundancy of frames in the most straightforward way.

Finally the more complex coder with the added motion compensated prediction increases a little more the performance, with 1 or more extra dB of PSNR in the low bitrate zone, even though the difference seems to become less noticeable with increasing Rate. Our guess is that this is not a limit of the motion compensation strategy. It is rather due to the very naive coding technique chosen. Using a block DCT transform and quantization the coefficients uniformly is suboptimal for a compression algorithm, and especially is not a good strategy for the residuals, which contain many high frequency components which are difficult to compress with a block DCT. This problem probably makes our third coder prefer the copy mode since it saves much more bandwidth and the quality that could be gained with motion compensation is not fully exploited. Thus we conclude the encoder number 3 is superior, but it could perform better if a different technique was chosen to compress the residuals or at least the DCT coefficients were compressed in a different way, not with uniform quantization.

BONUS

We repeated the experiments for the other included video (*Mother and Daughter*) and another control video made of always the same checkerboard frame, to check that the encoders were working properly.

Whereas in Foreman video there was quite some movement in the video, the Mother & Daughter video is relatively still, especially the background, which is almost completely unchanged between frames. This is noticeable in the PSNR vs Rate graph shown in Figure 5.1.

Figure 5 - Frame from Mother & Daughter video

Notice how much more Encoders 2 and 3 (intra+copy modes, and intra+copy+inter modes respectively) make use of inter frame redundancy, and achieve much higher compression with only minimal effect on PSNR

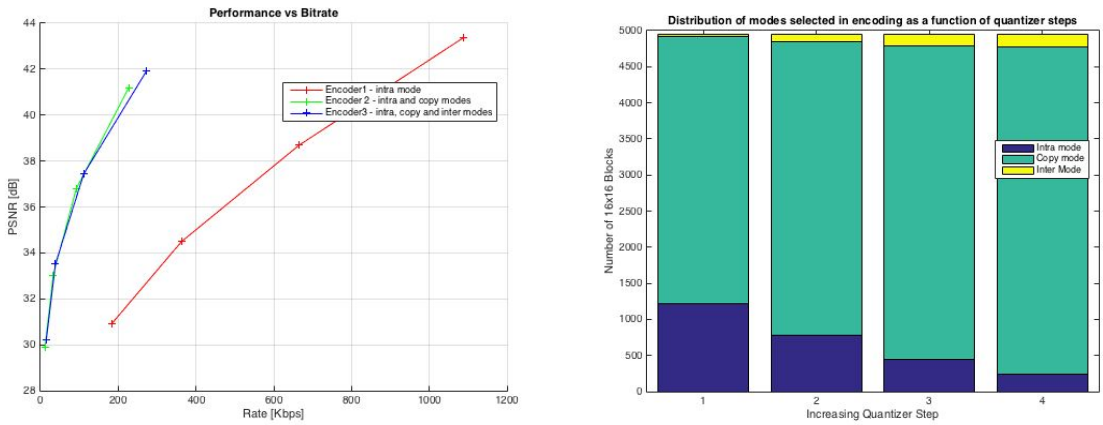

Figure 5.1 - Results for Mother & Daughter video. For a very stationary video, the copy mode is selected very often and encoder 2 and 3 thus achieve a much better performance compared to only intra mode

Finally we performed an extreme test case of compressibility, with a video consisting of always the same checkerboard frame. This was done to test that the encoders were working correctly. In this case the copy mode was always selected (except for frame 1 where only intra mode is possible) for both encoder 2 and 3, using always 1 bit/block and 2 bits/block respectively.

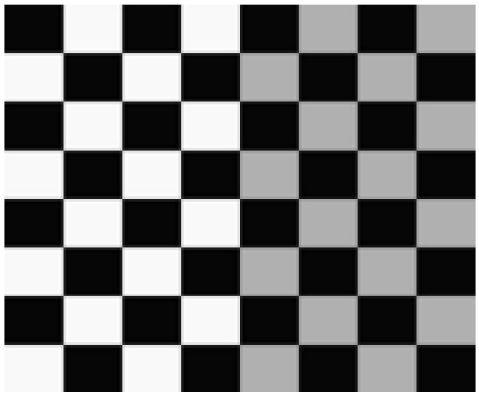

Figure 5.3 - The checkerboard test frame used to test the working of the two last encoders

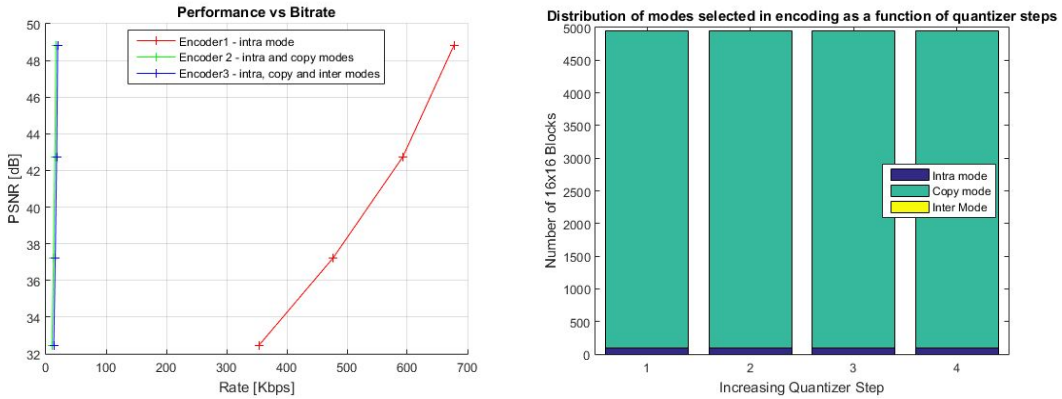

Figure 5.3 - Results for checkerboard constant video

SUMMARY

In this project, we investigated the performance of three encoders with increasing complexity. As expected, the two most complex coders outperformed the intra-frame coder, since it doesn't exploit at all the correlation between frames, which is one of the critical assets of video coding. Our results matched the theoretical gain of 6dB/bit of SNR when encoding with uniform quantization.

We also found a considerable reduction in bandwidth when exploiting the inter frame redundancy, especially for very time-stationary videos. Even if more complex, the motion compensated inter coder performs slightly better than the copy mode at the same level of compression and could work much better if the residual signal was encoded more efficiently or if the entropy of this last one was reduced further with more advanced motion compensation schemes (sub-pel, OBMC).

APPENDIX

The project and report was written collaboratively by both authors, while discussing the algorithms used throughout the process.
Part of the code mentioned in the report is shown below, and the remaining source code is attached in a zip folder. Please look in the zip folder also to find extra figures and videos.

Matlab code of some functions used

```
uniform_quantizer = @(x,ssize) round(x/ssize)*ssize;
mse = @(x,y) sum(sum((y-x).^2))/(size(y,1) * size(y,2));
PSNR = @(D) 10*log10(255^2./D);

function[Y] = dctz2(X)
%Discrete Cosine Transform of 8x8 block
%   X input matrix
%   Y output matrix
% Making sure it works for both matrix inputs and structs
if(isstruct(X))
    M = X.blockSize(1);
    X = X.data;
else
    M = size(X);
    M = M(1);
end

if (M~=8)
    error('unexpected block size, should be 8x8');
    Y = [];
    return
end
for k=0:M-1
    for i=0:M-1
        alpha = sqrt(((i~=0)+1)/M);
        c = cos(((2*k+1)*i*pi)/(2*M));
        A(i+1,k+1) = alpha * c; %stupid matlab non-zero indexing
    end
end
A = dctmtx(8);Y = A*X*A';
end
% EntropyRate
```

```
% Computes bits given a frame of quantized block DCT coefficients
% using information entropy to estimate the lower bound of the
% bits needed
%
% INPUT
% F - 8x8 blocked frame of quantized DCT coefficients
% step_q - integer step used to quantize the DCT coefficients in F
%
% OUTPUT
% rate - estimated bits needed to encode 1 dct coefficient
% it is equivalent to bits/pxl, since N_coeffs = N_pixels
%
function rate = EntropyRate( F,step_q )

video_height = size(F,1);
video_width = size(F,2);

coefs = zeros(8,8,(video_width/8)*(video_height/8));
index = 1;
ww = [1:8];
hh = [1:8];
%stack 8x8 blocks
for w=1:(video_width/8)
    for h=1:(video_height/8)
        coefs(ww,hh,index) = F(8*(h-1)+hh,8*(w-1)+ww);
        index = index+1;
    end
end
%now calculate R
H=zeros(8,8);
    for w=1:8
        for h=1:8
            vals = squeeze(coefs(h,w,:));  %coefs of frequency h,w
            p = hist(vals,min(vals):(step_q/32):max(vals));
            p = p/sum(p);
            % Entropy matrix
            H(h,w) = -sum(p.*log2(p+eps)); %eps added for log of 0 vals
        end
end
%average over all coefficients
rate = mean2(H);
end

% predFrame
%Predicts next frame from given frame and block motion vectors
% Input
%   imgI : The given frame, must be padded according to size of shifts
%   motionVect : The motion vectors in a [2 x N_blocks] matrix
%   bSize : Size of the block
%   shiftSize: vector containing [dy_max, dx_max]
%
% Ouput
%   predFrame : The motion compensated predicted frame
%
function predFrame = PredictFrame(F, motionVect, bSize, shiftSize)
```

```
dy_max = shiftSize(1);
dx_max = shiftSize(2);

[row, col] = size(F);

bCount = 1;
for i = 1+dy_max:bSize:row-bSize+1
    for j = 1+dx_max:bSize:col-bSize+1

        % dy is row(vertical) index
        % dx is col(horizontal) index
        % we are reading blocks line by line

        dy = motionVect(1,bCount);
        dx = motionVect(2,bCount);
        i_ref = i + dy;
        j_ref = j + dx;
        predF(i-dy_max:i-dy_max+bSize-1,j-dx_max:j-dx_max+bSize-1) = ...
            F(i_ref:i_ref+bSize-1, j_ref:j_ref+bSize-1);

        bCount = bCount + 1;
    end
end
predFrame = predF;
end

% Encoder3, with both copy conditional replenishment and motion compensation

% Rates per block are constants, to simplify
% +2 is added to signal the mode between the 3 available
% Rates are in Bits/block
R1 = mean(Rate1f,1)*bSize^2 + 2;         %intra mode
R2 = 2*ones(1,length(q_step));           %copy mode
R3 = mean(Rate3onlyf,1)*bSize^2 + 2;     %inter mode

Rates = [R1;R2;R3];

Mode123 = zeros(Nblocks,Nframes,length(q_step));
Mode123(:,1,:) = 1;  % cannot copy or inter code the first frame
Encoded3 = zeros(video_height,video_width,Nframes,length(q_step));
% Set first frame to intra coded - since it is the only coding possible for
% the first frame
for q=1:length(q_step)
    Encoded3(:,:,1,q) = Encoded1(:,:,1,q);
end
Nbits3 = zeros(Nframes,q);  %counts the bits used during the encoding
Nbits3(1,:) = R1*Nblocks;  %bits used for intra mode of 1st frame
% Loop over all frames and scan frames block by block
for f=1:Nframes-1
    for q=1:length(q_step)
        %Pad to take into account edges in motion vectors predictions
        Padded = padarray(Encoded3(:,:,f,q),[dy_max dx_max]);
        bCount = 1;
        ww = 1:bSize; %indices to get 16x16 blocks
        hh = 1:bSize;
        for h = 1:video_height/bSize
```

```
            for w = 1:video_width/bSize
                % For each mode, distortions for encoding the next block
                Diff1 = (Frames(bSize*(h-1)+hh,bSize*(w-1)+ww,f+1) - ...
                    Encoded1(bSize*(h-1)+hh,bSize*(w-1)+ww,f+1,q)).^2;
                D1 = sum(Diff1(:))/numel(Diff1(:));
                Diff2 = (Frames(bSize*(h-1)+hh,bSize*(w-1)+ww,f+1) - ...
                    Encoded3(bSize*(h-1)+hh,bSize*(w-1)+ww,f,q)).^2;
                D2 = sum(Diff2(:))/numel(Diff2(:));

                % Compute motion compensated coordinates
                dy = MotVecs(1,bCount,f);
                dx = MotVecs(2,bCount,f);
                y_compensated = bSize*(h-1) + dy + hh + dy_max;
                x_compensated = bSize*(w-1) + dx + ww + dx_max;
                Diff3 = (Frames(bSize*(h-1)+hh,bSize*(w-1)+ww,f+1) - ...
                    Padded(y_compensated,x_compensated)).^2;
                D3 = sum(Diff3(:))/numel(Diff3(:));

                % Encode next block with the mode that minimizes the
                % Lagrangian cost
                Cost1 = D1 + lambda123(q)*R1(q);
                Cost2 = D2 + lambda123(q)*R2(q);
                Cost3 = D3 + lambda123(q)*R3(q);
                % Select mode
                Costv = [Cost1,Cost2,Cost3];
                [MinCost,ChosenMode] = min(Costv);
                % Save mode history for visualization
                Mode123(bCount,f+1,q) = ChosenMode;
                % Accumulate Nbits used to get total Bits/Frame
                Nbits3(f+1,q) = Nbits3(f+1,q) + Rates(ChosenMode,q);

                % Encode video
                if ChosenMode == 1    %intra mode - set to Encoded1
                    Encoded3(bSize*(h-1)+hh,bSize*(w-1)+ww,f+1,q) = ...
                        Encoded1(bSize*(h-1)+hh,bSize*(w-1)+ww,f+1,q);
                elseif ChosenMode == 2  %copy block from previous frame
                    Encoded3(bSize*(h-1)+hh,bSize*(w-1)+ww,f+1,q) = ...
                        Encoded3(bSize*(h-1)+hh,bSize*(w-1)+ww,f,q);
                else   %inter mode, take previous shifted block + residual
                    Encoded3(bSize*(h-1)+hh,bSize*(w-1)+ww,f+1,q) = ...
                        Padded(y_compensated,x_compensated) + …
                           Residualq(bSize*(h-1)+hh,bSize*(w-1)+ww,f+1,q);
                end
                bCount = bCount+1;
            end
        end
    end
end
```

References

[1] *Rafael C. Gonzalez and Richard E. Woods*, Digital Image Processing, Prentice Hall, 2nd ed., 2002

[2] Everything about Data Compression! website with varioues resources and articles on video compression http://compression.ru/video/motion_estimation/index_en.html