Report for EQ2330 Image and Video Processing

EQ2330 Image and Video Processing, Project 3

Hanqi Yang hanqi@kth.se
Qian Zhou qianzho@kth.se

December 20, 2021

Summary

We implement intra-frame, conditional replenishment and motion compensation video coder in this project. The intra-frame video coder encodes each frame of a video sequence independently using DCT-based image coding from Project 2. Conditional replenishment video coder is the combination of intra mode and copy mode, and Lagrangian cost is used to decide which mode to choose. Motion compensation video coder introduces the inter mode based on the conditional replenishment coder, which includes motion-compensated prediction and residual coding. By using the rate-PSNR curve, we evaluate and compare their achievable performance.

1 Introduction

In this project, we implement three video coding algorithms which are intra-frame coder, conditional replenishment coder and motion compensation coder. We apply these algorithms to two videos “Foreman” and “Mother&Daughter” and evaluate their performance by measuring the rate-PSNR curves.

2 System Description

2.1 Intra-Frame Video Coder

An intra-frame coder encodes each frame of a video sequence independently from other frames. In this section, we subdivide the frame into 16×16 blocks and apply four two-dimensional 8×8 DCTs per 16×16 block. Then a scalar uniform quantizer is applied to all transform coefficients. By applying inverse DCTs to the quantized coefficients, we can obtain the reconstructed frames.
According to Parseval’s Theorem, the MSE between original and reconstructed image is equal to that between transform coefficients and quantized coefficients. Hence, we use the MSE of the latter one to calculate the PSNR. The approach in project 2 is applied to estimate the bit rate by calculating the average entropy of 50 frames.
The bit rate in unit of kbit/s should be measure as

$$bitrate = 30 \times height \times width \times entropy / 1000 \tag{1}$$

where *height* and *width* represent the size of the frame.

2.2 Conditional Replenishment Video Coder

In this section, we expand intra-frame coding by adding conditional replenishment.

There are two modes: copy mode and intra mode. The former one means the current 16 × 16 block is copied from the co-located 16 × 16 block in the previous frame if some blocks do not change much from frame to frame (for example, video background) and the latter one represents that the current block is intra-frame coding.
To decide whether to use the intra mode or the copy mode, we calculate the Lagrangian cost function

$$J_n = D_n + \lambda R_n \tag{2}$$

for each 16 × 16 block. D_n is the mean square error (MSE) distortion associated with mode n, and R_n is the bit rate associated with mode n. λ is proportional to the squared quantizer step-size Q^2. For better drawing results, we assume $\lambda = 0.002Q^2$. We set intra mode as mode 1 and copy mode as mode 2. D_1 is the MSE between the encoded block and the original block, and D_2 is the MSE between the copied block from the precious frame and the original block at the same location.
The bit rate of the copy mode is R_2 = 1 bit/block because 1 bit is needed to distinguish between the two modes. For intra mode, the bit rate is the entropy of the 16 × 16 block which can be calculated by

$$H = -\sum_{k=0}^{L-1} p_r(r_k) \times \log_2(p_r(r_k)) \tag{3}$$

$$R_1 = H \cdot blocksize^2 \tag{4}$$

where L is the number of intensity values, r_k is the intensities of a certain pixel and $p_k(r_k)$ is the probability of occurrence of r_k.
According to equation (2), we can choose the mode which achieves smaller cost value.

2.3 Video Coder with Motion Compensation

We introduce 16 × 16 block-based inter mode that allows block-based integer-pel motion-compensated prediction to expand the conditional replenishment video coder. We assume a displacement range of [-10,...,10] × [-10,...,10] pel for motion compensated prediction which means there are 21 possible shifts in both x and y direction. Hence, we need $2 \cdot \log_2 21 = 10$ bits to encode the motion vector and another 2 bits to select the three modes, for a total of 12 bits.
After using the motion compensation mode, the rest is encoded using the residual signal. The residual signal is the difference between the motion-compensated block and the original block, which is coded by intra-frame coder.
We still use equation (2) to calculate the Lagrangian cost of the three modes. We set inter mode as mode 3. The bit rate of mode 3 R_3 equals to 12 bits plus the bits for residual block coding. D_3 is the MSE between the original block and reconstructed block consisting of the residual block and the motion compensation block.

3 Results

3.1 Intra-Frame Video Coder

Figure 3-1 shows the rate- PSNR curves of “Foreman” and “Mother&Daughter” for intra-frame mode. Comparing the two curves, their trends are roughly the same. As the level of quantization increases, the bit rate increases which leads to an good performance in the video recovery. This conclusion can also be drawn from Figure 3-2. The frame with a quantization step of 8 is significantly clearer than that with step of 64. The slopes of the rate-PSNR curves at high bitrates of “Foreman” and “Mother&Daughter” are 0.0092 and 0.011, respectively.

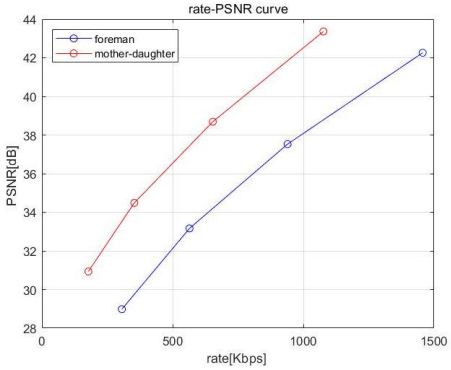


Figure 3-1: rate-PSNR curves for intra-frame video coder

(a)step = 8 (b) step = 16 (c)step = 32 (d)step = 64

Figure 3-2: Frames coded with intra-frame mode for different quantization steps

3.2 Conditional Replenishment Video Coder

Table 3.1 and 3.2 shows that the number of blocks using intra mode and copy mode simultaneously for different quantization step size. In both videos, as the quantized step size increases, copy mode is being used more frequently and the quality of frames decrease. Comparing the two tables, copy mode appears more in “Mother&Daughter” because the figures in this video do not move much in relation to the background.

The rate-PSNR curves for intra-frame video coder and conditional replenishment video coder of “Foreman” and “Mother&Daughter” are shown in Figure 3-3. For video “Foreman”, conditional replenishment coding requires slightly fewer bits than intra-frame coding at the same PSNR, i.e. better frame reconstruction with conditional replenishment coding at the same bit rate. For video “Mother&Daughter”, however, conditional replenishment coding performs much better than intra-frame coding as the bit rate using conditional replenishment coding is 300-500 kbit/s less than using

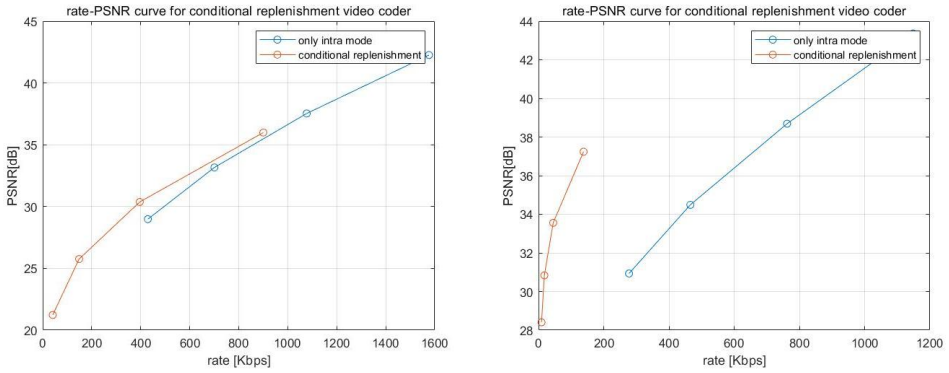
intra-frame coding at the same PSNR. In a word, for more still videos, conditional replenishment coding gives better reconstruction at the same bit rate.

Table 3.1: Number of blocks by two modes for different quantization steps (Foreman)

Step size	8	16	32	64
Intra mode	2620	1660	950	426
Copy mode	2330	3290	4000	4524

Table 3.2: Number of blocks by two modes for different quantization steps (Mother&Daughter)

Step size	8	16	32	64
Intra mode	469	236	148	114
Copy mode	4481	4714	4802	4836



(a)Foreman (b)Mother&daughter

Figure 3-3: rate-PSNR curves for conditional replenishment video coder

3.3 Video Coder with Motion Compensation

Table 3.3 and 3.4 shows that the number of blocks using intra mode, copy mode and inter mode simultaneously for different quantization step size. As with the conditional replenishment, as the step size increases, the frames carry less information, and thus the number of blocks using the copy mode increases. For both videos, the number of blocks using the inter mode is minimal.

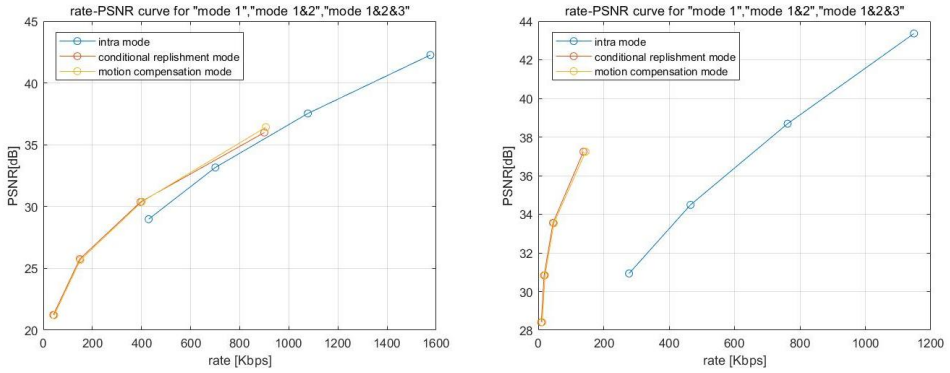
The rate-PSNR curves for intra-frame video coder, conditional replenishment video coder and motion compensation video coder of “Foreman” and “Mother&Daughter” are compared in Figure 3-4. There is not much difference in the performance of the latter two video coders in the two videos. The introduction of conditional replenishment and motion compensation has nicely improved the performance of the inter-frame, which is even more evident in the “mother&daughter” video because the “mother&daughter” figures do not move as much, i.e. the inter-frame blocks move less, allowing the motion compensation video coder to encode the video with fewer bits under the same PSNR.

Table 3.3: Number of blocks by three modes for different quantization steps (Foreman)

Step size	8	16	32	64
Intra mode	2296	1595	920	399
Copy mode	2167	3260	3965	4421
inter mode	487	95	65	130

Table 3.4: Number of blocks by three modes for different quantization steps (Mother&Daughter)

Step size	8	16	32	64
Intra mode	430	234	146	111
Copy mode	4429	4707	4793	4834
inter mode	91	9	11	5



(a)Foreman (b)Mother&daughter

Figure 3-4: rate-PSNR curves for three modes: “intra mode”, “intra mode and copy mode” and “intra mode, copy mode and motion compensation mode”

4 Conclusions

For all the three coders, as the level of quantization increases, the bit rate increases which leads to an good performance in the video recovery. For the conditional replenishment coder, as the quantized step size increases, copy mode is being used more frequently and the quality of frames decrease.

It is also worth to notice that the introduction of conditional replenishment mode and motion compensation mode improved video PSNR for the same bit rate. However, there is not much difference in the performance of conditional replenishment mode and motion compensation mode in the two videos. The latter one only achieves better PSNR than the former one for smaller quantization steps (better frame quality/lower frame compression).

In summary, the motion compensation video coder performs the best among all three tested modes. Conditional supplementation and motion compensation coding require fewer bits to reconstruct the image under the same PSNR condition, which is more effective for the more static video.

Reference

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

Appendix

Matlab Code

Main program: main_1.m, main_2.m, main_3.m
Function: yuv_import_y.m, yuv2mov.m, uniquant.m, shift.m, resbitrate.m, PSNR.m, mse.m, Lagrangian.m, dct16x16.m, idct16x16.m, bitrate.m.

Main program:

- **main_1.m**

```
% assignment 2
clc;
clear;

video_width = 176;
video_height = 144;
numfra = 50;
framerate = 30;
steplen = [3,4,5,6];
V1 = yuv_import_y('foreman_qcif.yuv',[video_width
video_height],numfra);
V2 = yuv_import_y('mother-daughter_qcif.yuv',[video_width
video_height],numfra);

% foreman
% DCT
for i = 1:length(V1)
    Vdct1{i,1} = blkproc(V1{i,1},[16 16],@dct16x16);
end

% Quantization

for i = 1:length(steplen)
    s = steplen(i);
    step = 2^s;
    for j = 1:numfra
        Vdctq1{j,i} = uniquant(Vdct1{j,1},step);
    end
end
```

```
% MSE
for i = 1:length(steplen)
    for j = 1:numfra
        error1(j,i) = mse(Vdct1{j,1},Vdctq1{j,i});
    end
end

% inverse DCT
for i = 1:length(steplen)
    for j = 1:numfra
        recon1{j,i} = blkproc(Vdctq1{j,i},[16 16],@idct16x16);
    end
end

for j=1:length(steplen)
    for i=1:numfra
        Video1(:,:,i,j) = recon1{i,j};
    end
end

Vpsnr1 = mean(PSNR(error1));

bitrates1 = zeros(size(steplen));
rownum = video_width/16;
colnum = video_height/16;
%calculate bitrates/s per each qunatizer step
for j=1:length(steplen) %for each quantizer step
    s = steplen(j);
    step = 2^s;
    K = zeros(16,16,rownum*colnum*length(Vdctq1));

    for i=1:length(Vdctq1) %for each frame
        index = 1;
        for row=1:rownum
            for col=1:colnum
                for m=1:16
                    for n=1:16
                        K(m,n,(i-1)*rownum*colnum+index) =
Vdctq1{i,j}(16*(col-1)+n,16*(row-1)+m);
                    end
                end
                index = index + 1;
            end
```

```
        end
    end

    % calculate R
    entropy=zeros(16,16);
    for m=1:16
        for n=1:16
            value = K(m,n,:);
            bins = min(value):step:max(value);
            pr = hist(value(:),bins(:));
            prb = pr/sum(pr);
            entropy(m,n) = -sum(prb.*log2(prb+eps)); %eps added for log
of 0 vals
        end
    end
    bitrates1(j) = mean2(entropy);
end
%covert to kbit/s
ratesKBPS1 = bitrates1 .*
((video_height*video_width*framerate)/1000);

% mother-daughter
% DCT
for i = 1:length(V1)
    Vdct2{i,1} = blkproc(V2{i,1},[16 16],@dct16x16);
end

% Quantization

for i = 1:length(steplen)
    s = steplen(i);
    step = 2^s;
    for j = 1:numfra
        Vdctq2{j,i} = uniquant(Vdct2{j,1},step);
    end
end

% MSE
for i = 1:length(steplen)
    for j = 1:numfra
        error2(j,i) = mse(Vdct2{j,1},Vdctq2{j,i});
    end
end
```

```
% inverse DCT
for i = 1:length(steplen)
    for j = 1:numfra
        recon2{j,i} = blkproc(Vdctq2{j,i},[16 16],@idct16x16);
    end
end

for j=1:length(steplen)
    for i=1:numfra
        Video2(:,:,i,j) = recon2{i,j};
    end
end

Vpsnr2 = mean(PSNR(error2));

bitrates2 = zeros(size(steplen));
rownum = video_width/16;
colnum = video_height/16;
%calculate bitrates/s per each qunatizer step
for j=1:length(steplen) %for each quantizer step
    s = steplen(j);
    step = 2^s;
    K = zeros(16,16,rownum*colnum*length(Vdctq2));

    for i=1:length(Vdctq2) %for each frame
        index = 1;
        for row=1:rownum
            for col=1:colnum
                for m=1:16
                    for n=1:16
                        K(m,n,(i-1)*rownum*colnum+index) =
Vdctq2{i,j}(16*(col-1)+n,16*(row-1)+m);
                    end
                end
                index = index + 1;
            end
        end
    end

    % calculate R
    entropy=zeros(16,16);
    for m=1:16
        for n=1:16
            value = K(m,n,:);
```

```
            bins = min(value):step:max(value);
            pr = hist(value(:),bins(:));
            prb = pr/sum(pr);
            entropy(m,n) = -sum(prb.*log2(prb+eps)); %eps added for log
of 0 vals
        end
    end
    bitrates2(j) = mean2(entropy);
end
%covert to kbit/s
ratesKBPS2 = bitrates2 .*
((video_height*video_width*framerate)/1000);

figure;
plot(ratesKBPS1, Vpsnr1, 'b-o');
hold on;
plot(ratesKBPS2, Vpsnr2, 'r-o');
title('rate-PSNR curve');
grid on;
legend('foreman','mother-daughter');
xlabel('rate[Kbps] ');
ylabel('PSNR[dB] ');
```

- **main_2.m**

```
% assignment 3
clc;
clear;

video_width = 176;
video_height = 144;
numfra = 50;
framerate = 30;
blksize = 16;
rownum = video_height/16;
colnum = video_width/16;
numblk = video_width*video_height/(blksize^2);
V = yuv_import_y('foreman_qcif.yuv',[video_width
video_height],numfra);
steplen = [3,4,5,6];

frame = zeros(video_height,video_width,numfra);
for f = 1:numfra
    frame(:,:,f) = V{f,1};
end
```

```
recon1 = zeros(video_height,video_width,numfra,length(steplen));
psnr1 = zeros(numfra,length(steplen));
rate1 = zeros(numfra,length(steplen));

recon2 = zeros(video_height,video_width,numfra,length(steplen));
psnr2 = zeros(numfra,length(steplen));
rate2 = zeros(numfra,length(steplen));

num_replblk = zeros(1,numfra,length(steplen));
num_replblk(:,1,:) = 0;

% code the first frame
for q = 1:length(steplen)
    m = 1:blksize;
    n = 1:blksize;
    step = 2^steplen(q);
    for row = 1:rownum
        for col = 1:colnum
            [recon2(blksize*(row-1)+m,blksize*(col-1)+n,1,q),rate]
= ...

bitrate(frame(blksize*(row-1)+m,blksize*(col-1)+n,1),step);
            rate2(1,q) = rate2(1,q) + rate;
        end
    end
    recon1(1,q) = recon2(1,q);
    rate1(1,q) = rate2(1,q);
    psnr2(1,q) = PSNR(mse(frame(:,:,1),recon2(:,:,1,q)));
    psnr1(1,q) = psnr2(1,q);
end

for f = 2:numfra
    for q = 1:length(steplen)
        m = 1:blksize;
        n = 1:blksize;
        step = 2^steplen(q);
        for row = 1:rownum
            for col = 1:colnum
                [blkrecon,R1] =
bitrate(frame(blksize*(row-1)+m,blksize*(col-1)+n,f),step);
                distortion1 =
mse(frame(blksize*(row-1)+m,blksize*(col-1)+n,f),blkrecon);
                distortion2 =
mse(frame(blksize*(row-1)+m,blksize*(col-1)+n,f),...
```

```
recon2(blksize*(row-1)+m,blksize*(col-1)+n,f-1,q));

            R1 = R1 + 1; % one additional bit for mode selection
            R2 = 1;

            J1 = Lagrangian(distortion1,step,R1);
            J2 = Lagrangian(distortion2,step,R2);
            J = [J1,J2];
            [minJ,choosemode] = min(J);

            if choosemode == 1 % intra mode
                recon2(blksize*(row-1)+m,blksize*(col-1)+n,f,q) =
blkrecon;
                rate2(f,q) = rate2(f,q) + R1;
            elseif choosemode == 2 % copy mode
                recon2(blksize*(row-1)+m,blksize*(col-1)+n,f,q)
= ...

recon2(blksize*(row-1)+m,blksize*(col-1)+n,f-1,q);
                num_replblk(1,f,q) = num_replblk(1,f,q)+1;
                rate2(f,q) = rate2(f,q) + R2;
            end
            recon1(blksize*(row-1)+m,blksize*(col-1)+n,f,q) =
blkrecon;
            rate1(f,q) = rate1(f,q) + R1 - 1;
          end
       end
       psnr1(f,q) = PSNR(mse(recon1(:,:,f,q),frame(:,:,f)));
       psnr2(f,q) = PSNR(mse(recon2(:,:,f,q),frame(:,:,f)));
   end
end

rate1 = mean(rate1,1)* framerate/1000;
rate2 = mean(rate2,1)* framerate/1000;
psnr1m = mean(psnr1,1);
psnr2m = mean(psnr2,1);
plot(rate1,psnr1m,'o-');
hold on;
plot(rate2,psnr2m,'o-');
grid on;
title('rate-PSNR curve for conditional replenishment video coder');
legend('only intra mode','conditional replenishment');
xlabel('rate [Kbps]');ylabel('PSNR[dB]');
```

```
% calculate the number of each mode
num_copymode = zeros(1,length(steplen));
num_total = zeros(1,length(steplen));
num_intramode = zeros(1,length(steplen));
for q = 1:length(steplen)
    for f = 1:numfra
        num_copymode(q) = num_copymode(q) + num_replblk(1,f,q);
    end
    num_total(q) = numblk*numfra;
    num_intramode(q) = num_total(q) - num_copymode(q);
end
```

- **main_3.m**

```
% assignment 4
clc;
clear;

video_width = 176;
video_height = 144;
numfra = 50;
framerate = 30;
blksize = 16;
rownum = video_height/16;
colnum = video_width/16;
numblk = video_width*video_height/(blksize^2);
V = yuv_import_y('mother-daughter_qcif.yuv',[video_width
video_height],numfra);
steplen = [3,4,5,6];

frame = zeros(video_height,video_width,numfra);
for f = 1:numfra
    frame(:,:,f) = V{f,1};
end

% initialize
recon1 = zeros(video_height,video_width,numfra,length(steplen));
psnr1 = zeros(numfra,length(steplen));
rate1 = zeros(numfra,length(steplen));

recon2 = zeros(video_height,video_width,numfra,length(steplen));
psnr2 = zeros(numfra,length(steplen));
rate2 = zeros(numfra,length(steplen));

recon3 = zeros(video_height,video_width,numfra,length(steplen));
```

```
psnr3 = zeros(numfra,length(steplen));
rate3 = zeros(numfra,length(steplen));

% intra, copy and motion compensation
num_interblk = zeros(length(steplen),3);

% motion compensation mode
maxshift = 10;
numshift = (2*maxshift + 1)^2;
shiftvector = zeros(2,numshift);
index = 1;
for dx = -maxshift:1:maxshift
    for dy = -maxshift:1:maxshift
        shiftvector(1,index) = dx;
        shiftvector(2,index) = dy;
        index = index + 1;
    end
end

% code the first frame
for q = 1:length(steplen)
    m = 1:blksize;
    n = 1:blksize;
    step = 2^steplen(q);
    for row = 1:rownum
        for col = 1:colnum
            [recon2(blksize*(row-1)+m,blksize*(col-1)+n,1,q),rate]
= ...

bitrate(frame(blksize*(row-1)+m,blksize*(col-1)+n,1),step);
            rate2(1,q) = rate2(1,q) + rate;
        end
    end
    recon1(:,:,1,q) = recon2(:,:,1,q);
    recon3(:,:,1,q) = recon2(:,:,1,q);
    rate1(1,q) = rate2(1,q);
    rate3(1,q) = rate2(1,q);
    psnr2(1,q) = PSNR(mse(frame(:,:,1),recon2(:,:,1,q)));
    psnr1(1,q) = psnr2(1,q);
    psnr3(1,q) = psnr2(1,q);
end

for f = 2:numfra
    for q = 1:length(steplen)
```

```
        m = 1:blksize;
        n = 1:blksize;
        step = 2^steplen(q);
        blknum = 1;
        shiftdirection =
shift(recon3(:,:,f-1,q),frame(:,:,f),shiftvector);

        for row = 1:rownum
            for col = 1:colnum

                % intra mode and copy mode
                [blkrecon,R1] =
bitrate(frame(blksize*(row-1)+m,blksize*(col-1)+n,f),step);
                distortion1 =
mse(frame(blksize*(row-1)+m,blksize*(col-1)+n,f),blkrecon);
                distortion2 =
mse(frame(blksize*(row-1)+m,blksize*(col-1)+n,f),...

recon2(blksize*(row-1)+m,blksize*(col-1)+n,f-1,q));
                R1 = R1 + 1;% one additional bit for mode selection
                R2 = 1;
                J1 = Lagrangian(distortion1,step,R1);
                J2 = Lagrangian(distortion2,step,R2);
                J = [J1,J2];
                [~,choosemode] = min(J);

                if choosemode == 1 % intra mode
                    recon2(blksize*(row-1)+m,blksize*(col-1)+n,f,q) =
blkrecon;
                    rate2(f,q) = rate2(f,q) + R1;
                elseif choosemode == 2 % copy mode
                    recon2(blksize*(row-1)+m,blksize*(col-1)+n,f,q)
= ...

recon2(blksize*(row-1)+m,blksize*(col-1)+n,f-1,q);
                    rate2(f,q) = rate2(f,q) + R2;
                end

                recon1(blksize*(row-1)+m,blksize*(col-1)+n,f,q) =
blkrecon;
                rate1(f,q) = rate1(f,q) + R1 - 1;

                % intra mode, copy mode and motion compensation mode
                dy = shiftvector(1,shiftdirection(blknum));
```

```
            dx = shiftvector(2,shiftdirection(blknum));
            ymov = blksize*(row-1) + dy + m;
            xmov = blksize*(col-1) + dx + n;
            movedBlk = recon3(ymov,xmov,f-1,q);
            [blkrecon_res, R3] =
resbitrate(movedBlk,frame(ymov,xmov,f),step);

            distortion3 =
mse(frame(blksize*(row-1)+m,blksize*(col-1)+n,f),blkrecon_res);

            R1 = 2 + R1; % two additional bit for mode selection
            R2 = 2;
            R3 = 2 + 10 + R3;
            J1 = Lagrangian(distortion1,step,R1);
            J2 = Lagrangian(distortion2,step,R2);
            J3 = Lagrangian(distortion3,step,R3);
            J = [J1,J2,J3];
            [minJ,choosemode] = min(J);

            if choosemode == 1 % intra mode
                recon3(blksize*(row-1)+m,blksize*(col-1)+n,f,q) =
blkrecon;
                rate3(f,q) = rate3(f,q) + R1;
            elseif choosemode == 2 % copy mode
                recon3(blksize*(row-1)+m,blksize*(col-1)+n,f,q)
= ...

recon3(blksize*(row-1)+m,blksize*(col-1)+n,f-1,q);
                num_interblk(q,2) = num_interblk(q,2) + 1;
                rate3(f,q) = rate3(f,q) + R2;
            elseif choosemode == 3 % motion compensation mode
                recon3(blksize*(row-1)+m,blksize*(col-1)+n,f,q) =
blkrecon_res;
                num_interblk(q,3) = num_interblk(q,3) + 1;
                rate3(f,q) = rate3(f,q) + R3;
            end
            blknum = blknum + 1;
         end
      end
      psnr1(f,q) = PSNR(mse(recon1(:,:,f,q),frame(:,:,f)));
      psnr2(f,q) = PSNR(mse(recon2(:,:,f,q),frame(:,:,f)));
      psnr3(f,q) = PSNR(mse(recon3(:,:,f,q),frame(:,:,f)));
   end
end
```

```
rate1 = mean(rate1,1)* framerate/1000;
rate2 = mean(rate2,1)* framerate/1000;
rate3 = mean(rate3,1)* framerate/1000;
psnr1m = mean(psnr1,1);
psnr2m = mean(psnr2,1);
psnr3m = mean(psnr3,1);
plot(rate1,psnr1m,'o-');
hold on;
plot(rate2,psnr2m,'o-');
hold on;
plot(rate3,psnr3m,'o-');
grid on;
title('rate-PSNR curve for "mode 1","mode 1&2","mode 1&2&3"');
legend('intra mode','conditional replishment mode','motion
compensation mode');
xlabel('rate [Kbps]');ylabel('PSNR[dB]');

num_total = zeros(1,length(steplen));
for q = 1:length(steplen)
   num_total(q) = numblk*numfra;
   num_interblk(q,1) = num_total(q) - num_interblk(q,2) -
num_interblk(q,3);
end
```

Functions:

- **uniquant.m**

```
function quant = uniquant(x,steplen)
% uniform mid-tread quantizer function
quant = round(x/steplen)*steplen;
end
```

- **PSNR.m**

```
function D = PSNR(d)
D = 10*log10(255^2./d);
end
```

- **mse.m**

```
function MSE = mse(x,y)
[m,n] = size(x);
MSE = sum((x-y).^2,'all')/(m*n);
end
```

- **dct16x16.m**

```
function blkdct = dct16x16(blk)
```

```
blkdct = blkproc(blk,[8 8],@dct2);
end
```

- **idct16x16.m**

```
function blkidct = idct16x16(blkdct)
blkidct = blkproc(blkdct,[8 8],@idct2);
end
```

- **Lagrangian.m**

```
function J = Lagrangian(D,Q,R)
% INPUT: D: mean square error distortion
%       Q: the squared quantizer step-size
%       R: the rate in bits
lamda = 0.002*Q^2;
J = D + lamda*R;
end
```

- **bitrate.m**

```
function [blkrecon,rate] = bitrate(blk,steplen)
[height,width] = size(blk);
blksize = 16;
blkdct = blkproc(blk,[8 8],@dct2);
blkdctq = uniquant(blkdct,steplen);
blkrecon = blkproc(blkdctq,[8 8],@idct2);
value = reshape(blkdctq(:,:),[1,height*width]);
bins = min(value):steplen:max(value);
pr = hist(value(:),bins(:));
prb = pr/sum(pr);
entropy = -sum(prb.*log2(prb+eps)); %eps added for log of 0 vals
rate = entropy*(blksize^2);
end
```

- **shift.m**

```
function bestvector = shift(preframe,curframe,shiftvector)
[height,width] = size(preframe);
blksize = 16;
rownum = height/blksize;
colnum = width/blksize;
MSE = zeros(rownum,colnum,length(shiftvector));
bestvector = zeros(1,rownum*colnum);

index = 1;
for i = 1:rownum
   for j = 1:colnum
      for d = 1:length(shiftvector)
         rows = 1+(i-1)*blksize:(i-1)*blksize + blksize;
```

```
            cols = 1+(j-1)*blksize:(j-1)*blksize + blksize;
            shiftrows = rows + shiftvector(1,d);
            shiftcols = cols + shiftvector(2,d);

if(shiftrows(1)<1||shiftcols(1)<1||shiftrows(blksize)>height||shi
ftcols(blksize)>width)
               MSE(i,j,d) = 99999999999999;
               continue;
            end
            MSE(i,j,d) =
mse(preframe(shiftrows,shiftcols),curframe(rows,cols));
       end
       vectors = find(MSE(i,j,:)==min(MSE(i,j,:)));
       if length(vectors)>1
          vectors = vectors(1,1);
       end
       bestvector(1,index) = vectors;
       index = index + 1;
   end
end
end
```

- **resbitrate.m**

```
function [recon,rate] = resbitrate(movblk,orgblk,steplen)
blksize = size(movblk,1);
[height,width] = size(movblk);
residual = orgblk - movblk;
resdct = blkproc(residual,[8 8],@dct2);
resdctq = uniquant(resdct,steplen);
resrecon = blkproc(resdctq,[8 8],@idct2);
recon = resrecon + movblk;
value = reshape(resdctq(:,:),[1,height*width]);
bins = min(value):steplen:max(value);
pr = hist(value(:),bins(:));
prb = pr/sum(pr);
entropy = -sum(prb.*log2(prb+eps)); %eps added for log of 0 vals
rate = entropy*(blksize^2);
end
```