Lab 6: Implementation of an operational video codec

In the last lab sessions, we will work toward the implementation of a simplified yet working video codec.

We will take advantage from all the previous labs to build up our working prototype.

Specifications

The final goal of these labs is to implement a gray-scale hybird video encoder, featuring:

- Hybrid video coding
- JPEG-like encoding in Intra mode
- Motion-compensated temporal prediction for Inter-Mode
- IPPP...P gop structure
- Predictive Exp-Golomb coding for motion vectors
- RD-optimized mode selection

The encoder will produce a binary bitstream and also the decoded images. These data are used to evaluate the Rate-Distortion performance of the encoder.

Use of global variables

As you might know, global variable can be very useful, but also source of errors. However, in this project it could be useful to have some global variables to reduce the parameter copy among functions and speed up the execution. Global variables could include the current image to be encoded, the decode frame buffer, and the set of coding parameters.

However, if you are not familiar with the use of global variables, just use normal variables.

Workflow

The workflow is the following

1. Lab no 6 (today): the goal is to understand the top-down view of the project and to start working on some simple module such as the image loader, the PSNR computation etc. Also, today we will use the blocks of previous labs in order to build an Intra-only video coder
2. The **Intra encoder** scans the blocks of the image and sends them to the encoding engine that is a block-based, JPEG-like encoding engine (INTRA coding mode for a block)
3. Next we will implement a block-level **temporal predictor** based on motion estimation and compensation (Lab no 7)
4. Lab no 7: now we can implement the Inter encoder that scans the blocks of an image; for each it tests both INTRA and INTER mode and makes an RD-optimized choice
5. Lab no 8: the goal is debugging and testing the encoder, and performing experiments, as for example: tracing rate-PSNR curves for different GOP sizes; comparing the effect of changing the search area for ME; comparing the effect of different values for λ (the optimization parameter for the mode decision) and for
6. Implementing the GOP encoder that encodes one image as INTRA and the following as INTER
7. Implementing the sequence encoder that works GOP by GOP and produces the encoding statistics

Top-down project description

Let us consider now a top-down view of the project, that should help in understanding the input/outputs of the different modules. We will use pseudo-code to describe the relationships among modules. Note that **not all the inputs/outputs are described in this high-level description**

Code for `experiments_script.m` (note: you have to copy/paste this code into a new Matlab file that you can then run).

```
%% Script for experiments %%

%% Initializations
% We use a struct to simplify the parameter exchange
global params img decodedFrameBuffer


% The variable params is a struct that contains all of the encoder configuration information
% It is a global, so it is available also within a function called in this script
% ** You have to properly fill those fields
params.IntraQualityFactorVector = 20; % Start with a single value, then use a vector for multiple encodings
deltaQ = 10;
params.InterQualityFactor = params.IntraQualityFactor-DeltaQ; % We use differe QF for Intra and Inter
params.inputVideoName = 'foreman'; % Use one of the provided .y files
params.inputVideoFile   = [params.inputVideoName '_cif.y']   % full input file name. Add the path if needed
params.modeSelectionLabmda = 10;  % Lagrangian param for mode selection. Start with 10
params.motionEstimationLambda = 3; % Lagrangian param for ME. Start with 3
```

```
params.ME.search=5;  % ME search window size
params.ME.dist = 'SAD'; % ME metric
params.GOPSize = 1; % Start with an "All Intra" configuration. Then move to IPP..P  Gops
params.numberOfGops = 3; % Start with a small number of GOPS for test, then use the full sequence
params.blockSize = 8;  % Can be modified later
params.nRow = 288;
params.nCol = 352;
params.verbose = 1;  % Setting the verboseness level might be useful for debugging and testing
% other possible parameters can be set as fields of the struct
decodedFrameBuffer.reference = zeros(params.nRow,params.nCol); % Since we do not use B, one decoded refernce is enough
decodedFrameBuffer.current   = zeros(params.nRow,params.nCol); % We also need a buffer for the current encoded image
params.compressedFileName = 'test.bin';


%% Initializations
% We want to perform a number pf encoding of a video sequence
% This number is the number of quantation steps
% For each quantization step, we compute a rate and a distortion
% Let us create the data structure to store these pieces of information

nEncodings = numel(params.IntraQualityFactorVector);
PSNR = zeros(nEncodings, 1);
RATE = zeros(nEncodings, 1);

%% Init for compression
params.q_mtx1 =     [16 11 10 16 24 40 51 61;
            12 12 14 19 26 58 60 55;
            14 13 16 24 40 57 69 56;
            14 17 22 29 51 87 80 62;
            18 22 37 56 68 109 103 77;
            24 35 55 64 81 104 113 92;
            49 64 78 87 103 121 120 101;
            72 92 95 98 112 100 103 99];


%% Main loop
for qualityIndex = 1:n
    param.qual = params.IntraQualityFactorVector(qualityIndex);
    [R, P] = sequenceEncoder(); %don't need input arg because param is global!
    RATE(qualityIndex) = R;
    PSNR(qualityIndex) = P;
end

%% This main loop can be replicated in order to test and compare different configurations, for example
params.GOPSize = 2; % let us test an IPIP... gop structure
PSNR2 = zeros(nEncodings, 1);
RATE2 = zeros(nEncodings, 1);
%% Main loop
for qualityIndex = 1:n
    param.qual = params.IntraQualityFactorVector(qualityIndex);
    [R, P] = sequenceEncoder(); %don't need input arg because param is global!
    RATE2(qualityIndex) = R;
    PSNR2(qualityIndex) = P;
end

% Compare the results
plot(RATE, PSNR,RATE2, PSNR2) ;
legend('All-Intra', 'IPIP gop');
```

Now let us consider the `sequenceEncoder` function: it must encode a GOP, retrieve its rate and PSNR, and update the sequence rate and PSNR.

Code for `sequenceEncoder.m` (copy and past to a file)

```
function [RATE PSNR] = sequenceEncoder()
% This function encodes a full sequence
% All the encoder configuration is specified in params
% It returns the coding rate (bpp) and the PSNR

global params img decodedFrameBuffer

% Initialization: how many images - and store their rate and PSNR
nFrames = params.GOPSize * params.numberOfGops;
perFramePSNR = zeros(nFrames, 1);
perFrameRATE = zeros(nFrames, 1);

% define the quantization matrix
SF = (params.qual==100)+(params.qual<=99&&params.qual>50)*(200-2*params.qual)+ (params.qual<=50)*(5000/params.qual);
q_mtx = ceil(params.q_mtx1*SF/100);
params.q_mtx (q_mtx>256)=256;
```

```
% Create and open the output file
params.fid = fopen(params.compressedFileName, 'wb');

% Main loop
for gopIndex = 1:params.numberOfGops,
     params.firstFrameGOP =  params.GOPSize*(gopIndex-1) + 1;
     params.lastFrameGOP  =  params.GOPSize*gopIndex;
     [GOP_RATE, GOP_PSNR] = gopEncoder(params, gopIndex);
     perFramePSNR(params.firstFrameGOP:params.lastFrameGOP)   =  GOP_PNSR;
     perFrameRATE(params.firstFrameGOP:params.lastFrameGOP)   =  GOP_RATE;
end

% Compute average rate and PSNR
PSNR = mean(perFramePSNR);
RATE = mean(perFrameRATE);

% Close the output file
fclose(params.fid);
```

The `gopEncoder` function shall encode a full GOP

Code for `gopEncoder.m` (copy and past to a file)

```
function [GOP_RATE, GOP_PSNR] = gopEncoder(gopIndex)
% This function encodes a full GOP
% It updates the compressed file
% It computes the per-frame rate and PSNR
global params img decodedFrameBuffer

% First image in the GOP: INTRA
img = loadIntra(gopIndex) % This function loads a single Y image from the input file
imgBitStream = IntraImgEncoder(); % This function encodes a single image in Intra mode
% write the encoded intra image to file
appendToFile(imgBitStream);
% Update rate and psnr
GOP_PSNR(1) = ypsnr(img, decodedImageBuffer.current); % ypsnr is a function that shall compute the PSNR between Y images
GOP_RATE(1) = numel(imgBitStream / (params.nRow * params.nCol) ; % rate in bpp
% the currently decoded image becomes the reference for the next frame
decodedFrameBuffer.reference = decodedFrameBuffer.current;

% Other images: INTER
for interIndex = 1:numberOfInter
   img = loadInter(gopIndex,interIndex);
   [imgBitStream, imgPSNR, decodedImg] = InterImgEncoder();

    % write the encoded inter image to file
    appendToFile(imgBitStream);
    % Update rate and psnr
    GOP_PSNR(interIndex+1) = ypsnr(img, decodeImageBuffer.current);
    GOP_RATE(interIndex+1) = numel(imgBitStream / (params.nRow * params.nCol) ;
    % the currently decoded image becomes the reference for the next frame
    decodedFrameBuffer.reference = decodedFrameBuffer.current;

end
```

Now we can start implementing functions. We need the following:

`loadIntra.m`

```
function loadedImg = loadIntra(gopIndex)
```

This function loads the first image of gop number `gopIndex` (starting from 1), from the file `params.inputVideoFile`

`appendToFile.m`

```
function appendToFile(imgBitStream)
```

This function appends the current encoded image to the compressed output file

`ypsnr.m`

```
function psnr = ypsnr(img1,img2)
```

This function computes the PSNR between two luminance-only images

QUESTION 1.

The first goal of this lab is to complete these three functions. Open the provided files and complete them one by one. Use the script `test_functions.mlx` and follow the instructions.

Block-based INTRA coding

Now we have to implement the full encoding of an INTRA image.

We will use 2 functions: `IntraImgEncoder.m` scans the blocks of the image, while `blockIntraCoding.m` performs the encoding of the single block. Here we can re-use the code from lab no 4 to perform the JPEG-like encoding of a block.

`IntraImgEncoder.m`

```
function imgBitStream = IntraImgEncoder()
```

This function encodes the image stored in the global variable `img` and uses the global variable params to configure the encoding. The decoded image is stored in the global variable `decodedFrameBuffer.current`

The function returns the encoded bitstream.

`blockIntraCoding.m`

```
function [blockBitStream, decodedBlock, QDC] = blockIntraCoding(currentBlock,previousDC)
```

This function encode a single block. It performs 2D-DCT, quantization and lossless coding. Moreover it decodes the block and return it. It also return the bitstream of the encoded block and the quantized DC coefficient, that will be used to encode the next block. Here you should reuse as much as possible the material provided in lab 4.

Question 2. Complete the `IntraImgEncoder` and `blockIntraCoding` functions. Use the `test_functions.m` file to test them. You can compare with the results of Lab4. The bitstream produced should be the same. Also the rate and PSNR should be the same.

Lab 7

In this lab we continue working on the video encoder implementation.

Now we can use the function completed last week and run the video encoder in Intra only configuration.

In order to do that, create the files `sequenceEncoder.m` and `gopEncoder.m` as described in lab 6, and use the following to test the video codec in intra only configuration:

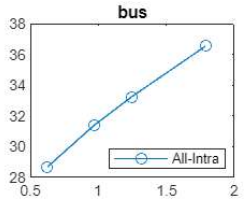
```
clear variables
global params img decodedFrameBuffer

params.IntraQualityFactorVector = 20:20:80;
params.deltaQ = 10; % The QF of Inter frame is  smaller than that of INTRA
params.inputVideoName = 'bus'; % Use one of the provided .y files
params.inputVideoFile   = [params.inputVideoName '_cif.y']   % full input file name. Add the path if needed
```

```
params = struct with fields:
           quantizationSteps: 20
              inputVideoName: 'bus'
              inputVideoFile: 'bus_cif.y'
          modeSelectionLabmda: 10
       motionEstimationLambda: 3
                          ME: [1×1 struct]
                     GOPSize: 1
                   blockSize: 8
                        nRow: 288
                        nCol: 352
                     verbose: 1
                firstFrameGOP: 11
                         fid: 5
                        qual: 80
                      q_mtx: [8×8 double]
```

```
params.GOPSize = 1; % Start with an "All Intra" configuration. Then move to IPP..P  Gops
params.numberOfGops = 3; % Start with a small number of GOPS for test, then use the full sequence
params.blockSize = 8;  %
params.nRow = 288;
params.nCol = 352;
params.verbose = 0;  % Setting the verboseness level might be useful for debugging and testing
% other possible parameters can be set as fields of the struct
decodedFrameBuffer.reference = zeros(params.nRow,params.nCol); % Since we do not use B, one decoded refernce is enough
decodedFrameBuffer.current   = zeros(params.nRow,params.nCol); % We also need a buffer for the current encoded image
params.compressedFileName = 'test.bin';

nEncodings = numel(params.IntraQualityFactorVector);
PSNR = zeros(nEncodings, 1);
RATE = zeros(nEncodings, 1);

%% Init for compression
params.q_mtx1 =     [16 11 10 16 24 40 51 61;
            12 12 14 19 26 58 60 55;
            14 13 16 24 40 57 69 56;
            14 17 22 29 51 87 80 62;
            18 22 37 56 68 109 103 77;
            24 35 55 64 81 104 113 92;
            49 64 78 87 103 121 120 101;
            72 92 95 98 112 100 103 99];


%% Main loop
for qualityIndex = 1:nEncodings
    QF = params.IntraQualityFactorVector(qualityIndex)
    params.qual =  QF;
    [R, P] = sequenceEncoder(); %don't need input arg because param is global!
    RATE(qualityIndex) = R;
    PSNR(qualityIndex) = P;
end
```

```
QF = 20
QF = 40
QF = 60
QF = 80
```

```
% Show the results
plot(RATE, PSNR, 'o-') ;
title(params.inputVideoName)
legend('All-Intra','Location','southeast');
```



You should obtain rates around 0.4 to 1.2 bpps and PSNR in the range 30-40 dB for "foreman" and similar values for the other sequences.

Motion Estimation

The core of the Inter coding step lies in the motion estimation function.

We want to implement a motion estimation function with the following features:

The input is a **block** of the current image. The coordinates of the block are stored in the global variable `params`.

There are two output parameters: the best-matching **block** (i.e., the *predictor* of the current block) and the estimated motion vector.

Within the motion estimation function, we use the global variable `decodedFrameBuffer.reference` to access to the reference image. We also use a new global variable, `MVF`, which is useful to store the already estimated motion vector.

The ME criterion is a regularized SAD. The regulatization term is the coding cost of the motion vector. The motion vector is encoded with spatial prediction and signed Exp-Golomb. **Notice that this is the same ME technique that we used in Lab. no 5.**

All the parameters of the motion estimation are stored in the global variable `params.ME`

Here we provide: first the skeleton code for the motion estimation function; and second, a testing environment for the function.

QUESTION 1. Complete the code of the `motionEstimation.m` function (remember to copy/past the following code in a new file, and then complete the missing parts). Test the function in the next environment.

```
function [bestMatch, mv] = motionEstimation(currentBlock)
% This function looks for the best match to currentBlock in the image
% stored in the global variable decodedFrameBuffer
% It returns the best matching block and the motion vector
% The cost function is a regularized SAD
% The regularization parameter is the coding cost of the motion vector

global params decodedFrameBuffer MVF

%% Initializations
% look up the block position and size
colIndex = params.colIndex;
rowIndex = params.rowIndex;
blockSizeRows = params.blockSize;
blockSizeCols = params.blockSize;

% look up the Lagrangian parameter for ME
lambda = params.ME.lambda;

% look up the image size
ROWS = params.nRow;
COLS = params.nCol;

% look up the search radius
radius = params.ME.radius;

% Initialization of the best displacement
bestDeltaCol=0; bestDeltaRow=0;
% Best cost initialized at the highest possible value
Jmin=blockSizeRows*blockSizeCols*256*256;
% Init the matching block
bestMatch = zeros(blockSizeRows,blockSizeCols);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Motion vector prediction %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% For the current position, we find the predictor of the motion
% vector. This predictor is used to encode the MV

% 1. There is no predictor for the first block: in this case
% we use (0,0)
if colIndex==1 && rowIndex ==1
    predictor = [0; 0];
    %Since there is no predictor in this case, we set
    %the penalization to zero
    weight = 0;
    % 2. For the first column (left), we take as predictor
    % the top neighbor. In this and all the other cases, a
    % predictor exists, so the penalization weight is lambda
elseif colIndex==1
    predictor = squeeze(MVF(rowIndex-blockSizeRows, colIndex,:));
    % The squeeze function makes sure that what we extract from the
    % 3D matrix MVF is, indeed, a column vector
    weight = lambda ;
    % 3. For the first row, we use the left neighbor
elseif rowIndex==1
    predictor = squeeze(MVF(rowIndex, colIndex-blockSizeCols,:));
    weight = lambda ;
else
    % 4. In all the other cases we take the MEDIAN of 3
    % neighbors: 1. the left neighbor, 2. the top neighbor
    V1 = squeeze(MVF(rowIndex, colIndex-blockSizeCols, :));
    V2 = squeeze(MVF(rowIndex-blockSizeRows, colIndex, :));
    % The third neighbor is the top-right neighbor if it is
    % available (ie., except for the last column)
    if colIndex<(COLS-blockSizeCols)
        V3 = squeeze(MVF(rowIndex-blockSizeRows, colIndex+blockSizeCols, :));
    else
        % For the last column we take the top-left neighbor
        V3 = squeeze(MVF(rowIndex-blockSizeRows, colIndex-blockSizeCols, :));
    end
    % 5. Computing the median: the three neighbors are put as
    % column of a matrix, and then the median is computed row-wise
    predictor = median([V1,V2,V3],2);
    weight=lambda;
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Motion Estimation main loop %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% ME loop on candidate motion vectors v = (deltaCol,deltaRow)
% It is a full search in [-radius:radius]^2
for deltaCol= ??? %% To be completed
    for deltaRow= ??? %% To be completed
        % Check: the candidate vector must point inside the image
        if ((rowIndex+deltaRow>0)&&(rowIndex+deltaRow+blockSizeRows-1<=ROWS)&& ...
                (colIndex+deltaCol>0)&&(colIndex+deltaCol+blockSizeCols-1<=COLS))
            % Now we are sure that the motion vector points inside
            % the image and we can recover the reference block R
            % Notice that R is obtained by adding the suitable
            % diplacement to the row and col indexes
            R=decodedFrameBuffer.reference(rowIndex+???:rowIndex+???+blockSizeRows-1, ...
                colIndex+???:colIndex+???+blockSizeCols-1);
            differences =  ??? ;
            SAD= ???;

            %% Regularization

            % 2. The regularization cost is the coding cost of the
            % prediction error

            predErr = [deltaRow; deltaCol]-predictor;
            cw = [encodeSEG(??? ), encodeSEG(???)];

            bits = numel(cw);

            J= SAD + weight*bits;

            % If current candidate is better than the previous
            % best candidate, then update the best candidate
            if (J<Jmin)
                Jmin=J;
                bestDeltaCol=deltaCol;
                bestDeltaRow=deltaRow;
                bestMatch = R;
            end

        end %
    end %
end % loop on candidate vectors
% Store the best MV
mv(1) = bestDeltaRow;
mv(2) = bestDeltaCol;
```

The `motionEstimation` function can be tested in the following environment.

QUESTION 2. Open the provided `loadInter.m` function, and understand its operation. Describe precisely its input and output parameters.

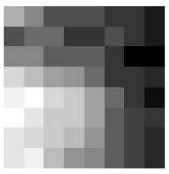
```
clear variables
global params img decodedFrameBuffer MVF

% Setting the configuration
params.inputVideoName = 'flower'; % Use one of the provided .y files
params.inputVideoFile   = [params.inputVideoName '_cif.y'];   % full input file name. Add the path if needed
params.blockSize = 8;  %
params.nRow = 288;
params.nCol = 352;
params.verbose = 2;  % Setting the verboseness level might be useful for debugging and testing
decodedFrameBuffer.reference = zeros(params.nRow,params.nCol); % Since we do not use B, one decoded refernce is enough
decodedFrameBuffer.current   = zeros(params.nRow,params.nCol); % We also need a buffer for the current encoded image
MVF = zeros(params.nRow,params.nCol,2);
params.ME.lambda = 10;
params.ME.radius = 20;

% Let us load two images from GOP 1
gopIndex = 1;
params.GOPSize = 10;
interIndex = 5;
params.firstFrameGOP =  params.GOPSize*(gopIndex-1) + 1;
img = loadIntra(gopIndex);
```

```
Loading Intra image from GOP 001 (image no 001 in the file)
```

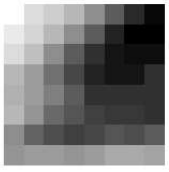
```
ref =  loadInter(gopIndex,interIndex);
```

```
Loading Inter image   5 from GOP 001 (image no 006 in the file)
```

```
%figure; imagesc([img;ref]); colormap("gray"); axis image; axis off;
% For the sake of simplicity, we put the original ref in the
% decodedFrameBuffer
decodedFrameBuffer.reference= ref;

% let us take a block from the current image
params.colIndex = 164; params.rowIndex=129;
currentBlock = img(params.rowIndex:params.rowIndex+7,params.colIndex:params.colIndex+7);
figure; imagesc([currentBlock]); colormap("gray"); axis image; axis off;
```



```
[bestMatch, mv] = motionEstimation(currentBlock);
figure; imagesc([bestMatch]); colormap("gray"); axis image; axis off;
```



```
fprintf('The best MV is (%d,%d)\nThe PSNR is %5.2f\n', mv(1),  mv(2), ypsnr(currentBlock,bestMatch));
```

```
The best MV is (1,18)
The PSNR is 29.67
```

QUESTION 3. Try the motion estimation with several blocks/images/sequences. As a reference when we take images 1 and 6 from the video sequence "flower" , for the block in position 164,129 from the sequence "flower", the best vector is (1,-4), with a PSNR=24.32 dB.

Test other blocks and positions.

akiyo

```
The best MV is (0,0)
The PSNR is 46.37
```

bus

```
The best MV is (7,17)
The PSNR is 18.94
```

foreman

```
The best MV is (1,18)
The PSNR is 29.67
```

Inter block coding

Now we can implement the function that encodes a block in Inter mode. We can take inspiration from the `blockIntraCoding` function, and suitably adapt it to the case of Inter coding.

Now we have to implement the `blockInterCoding` function. Here we must first find the best temporal predictor by using motion estimation; then we have to encode the prediction error in a JPEG-like fashion. Here you can find the code to complete in order to achieve the function. As usual, copy past in a new file, and complete the missing parts.

QUESTION 4. Complete the `blockInterCoding.m` function (copy/past into a new file, complete the missing parts) and test it in the provided test environment.

```
function [blockBitStream, decodedBlock, QDC, mv] = blockInterCoding(currentBlock,previousDC)
global params

% Motion estimation, producing the predictor and the motion vector
[bestMatch, mv] = motionEstimation(???);

% Encode the motion vector
blockBitStream = [encodeSEG(???), encodeSEG(???)];

% Compute the prediction error
predictionErrorBlock = ???;

% Encode the prediction error as it was INTRA coding
dctBlock = dct2(???);
% A special quantization matrix must be used
quantizedDctBlock = round(dctBlock./params.q_mtx_Inter);
%-----------------------------------
if params.verbose==2  % We display this only if verbose is on level 2
    disp('Original block:')
    currentBlock
    disp('Best predictor block:')
    bestMatch
    disp('Motion vector')
    mv
    disp('DCT')
    dctBlock
    disp('Quantized DCT')
    quantizedDctBlock
end
%-----------------------------------
losslessCoderVerboseness = 0;
blockBitStream = block_entropy_coding(???);
invQuant = params.q_mtx_Inter .* quantizedDctBlock;
decodedPredErrBlock = idct2(???);
QDC = quantizedDctBlock(1,1);

% The decoded block is obtained by adding the quantized prediction error to the matched block
decodedBlock = ??? ;

%-----------------------------------
if params.verbose==2
    disp('Inv. Quant. ')
    invQuant
    disp('Decoded')
    decodedBlock
end
%-----------------------------------
```

Let us test the block inter coding function.

```
clear variables;
clc
global params img decodedFrameBuffer MVF

% Setting the configuration
params.inputVideoName = 'akiyo'; % Use one of the provided .y files
params.inputVideoFile   = [params.inputVideoName '_cif.y'] ;  % full input file name. Add the path if needed
params.blockSize = 8;  %
params.nRow = 288;
params.nCol = 352;

decodedFrameBuffer.reference = zeros(params.nRow,params.nCol); % Since we do not use B, one decoded refernce is enough
decodedFrameBuffer.current   = zeros(params.nRow,params.nCol); % We also need a buffer for the current encoded image
MVF = zeros(params.nRow,params.nCol,2);
params.ME.lambda = 10;
params.ME.radius = 20;

% Let us load two images from GOP 1
gopIndex = 1;
params.GOPSize = 10;
interIndex = 1;  % The first inter frame
params.firstFrameGOP =  params.GOPSize*(gopIndex-1) + 1;
intra = loadIntra(gopIndex);
img =  loadInter(gopIndex,interIndex);

% For the sake of simplicity, we put the original intra frame (and not a decoded version) in the
% decodedFrameBuffer
decodedFrameBuffer.reference= intra;

% let us take a block from the current image
params.colIndex = 164; params.rowIndex=129;
currentBlock = img(params.rowIndex:params.rowIndex+7,params.colIndex:params.colIndex+7);
% Let us set an arbitrary value of the previous DC coeff
previousDC = 0;
params.verbose = 0;
[blockBitStream, decodedBlock, QDC, mv]= blockInterCoding(currentBlock,previousDC);

fprintf('The block is INTER-encoded using %d bits, ie %5.4f bpp.\n', numel(blockBitStream),numel(blockBitStream)/params.blockSize/par
```

```
The block is INTER-encoded using 6 bits, ie 0.0938 bpp.
```

```
fprintf('The PSNR is %5.3f\n', ypsnr(currentBlock,decodedBlock));
```

```
The PSNR is 48.339
```

```
fprintf('The best MV is (%d,%d)\n', mv(1),  mv(2));
```

```
The best MV is (0,0)
```

```
% Let us compare with the Intra coding of the same block
params.verbose = 0;
[blockBitStream, decodedBlock, QDC]= blockIntraCoding(currentBlock,previousDC);

fprintf('The block is INTRA-encoded using %d bits, ie %5.4f bpp.\n', numel(blockBitStream),numel(blockBitStream)/params.blockSize/par
```

```
The block is INTRA-encoded using 60 bits, ie 0.9375 bpp.
```

```
fprintf('The PSNR is %5.3f\n', ypsnr(currentBlock,decodedBlock));
```

```
The PSNR is 44.188
```

Try different sequences and images. In general, Inter coding should be more effective than Intra. But for some cases (i.e., new areas that cannot be predicted), Intra could be better. Try for example with images that are relatively temporally far apart.

	INTER-encoded PSNR	INTER-encoded best MV	INTRA-encoded PSNR
foreman	41.328	(7,-15)	42.163
flower	32.383	(0,1)	31.097
bus	35.229	(0,-7)	34.316
akiyo	48.339	(0,0)	44.188

For flower , bus and akiyo inter is better. But for foreman , intra is better.

Finalization of the video encoder

Inter image coding

In the previous labs, we have implemented the INTRA and INTER coding for a single block, as well as the INTRA coding for a whole image. Now the focus is on Inter coding of the whole image. We start by listing the features that should be supported by the function performing the INTER coding; then a partial implementation of the function is provided. **The first goal of this lab is** to understand and complete it. Finally, the new function is integrated into the full video codec. **The second goal** is to use the complete video encoder to perform some experiments: a list of possibilities is provided. **In the final report you should provide the results of the experiments and, most importantly, your comments.**

Features of the Inter Image coding function

Image pre-processing

Contrary to what we did for INTRA images, in the proposed implementation we do not subtract 128 to the image before starting the encoding process. This choice simplifies somewhat the encoding process.

Rate-distortion omptimized mode selection (RDO-MS)

An important feature of this function is that it should implement the **rate-distortion optimized mode selection (RDO-MS)**. What does it means? It means that each block must *first* be encoded in both INTRA and INTER mode; *then*, the encoding function evaluate the RD cost $J = D + \lambda R$ and choose the mode that minimizes the cost. The parameter λ is stored in the `params` global variable. We notice that this parameter has a similar, but different role than λ_{ME}: λ allows to weights between the coding rate and the MSE of a block; λ_{ME} allows to weight between the coding rate of a *motion vector* and the MSE of the predictor. However, in both cases, we should use a large value of the lagrangian parameter if the available bit rate is small. In practice, the values of the parameters are chosen empirically as a function of the quality factor. In our implementation, we set a fixed value of λ and set $\lambda_{\text{ME}} = \sqrt{\lambda}$ (which is common practice).

Coding cost maps `map.JIntra` and `map.JInter`

In order to collect some statistics about the mode selection the `IntraIngEncoder` function will return the coding costs of all the blocks, both for Intra and Inter mode. In this way, we can have further insight about the relationship between the content and the mode decision. This information is stored in the output variable `map`, which in turn has two fields: `map.JIntra` and `map.JInter`. Each one is a matrix with one entry per image block. Thus, each map's number of rows or columns is obtained from the image number of rows or columns divided by the block size. The value of each entry is the coding cost computed with the respective mode.

The Motion Vector Field `MVF`

Using RDO-MS implies that each block of the INTER image can be encoded either in INTRA or in INTER mode. However, in order to encode the motion vector of an INTER block, we need the motion vectors of its neighboring blocks. We store all the computed motion vectors in a global variable named `MVF`.

How to cope with the case when one or more of the neighboring blocks are encoded in INTRA mode? they will not have an associated motion vector. A simple solution folows: both the encoder and the decoder agree that, when a block is encoded as INTRA, it has a "virtual" motion vector equal to (0,0). This virtual neighbor is used in the MV prediction process as it was a "real" one. In conclusion, we just need to be sure that, corresponding to INTRA-coded blocks, the motion vector field is null.

Note that other, more efficient but more complicated solutions could be used (and actually are in modern video encoders), but the one proposed here is a reasonable compromise between complexity and efficiency.

DC prediction

When we encode a blockof the INTER image as an INTRA block, we are not sure that the previous block has been encoded as INTRA as well. So the question is: how the perform DC prediction? Here we take a simple choice: we use the last DC coefficient from an INTRA block as predictor, and the predictor for the first INTRA block is set to 128 (remember that we did not remove 128 from the image in the preprocessing step).

On the other hand, DC prediction makes less sense for INTER blocks, because the DCT is computed over a prediction residual. Therefore, we will call the `blockInterCoding` function with a conventionally zero value for the DC predictor, and we will not use the quantized DC coefficient produced by the DCT on the INTER block.

Signalization

We notice that the INTER/INTRA decision must be signalled to the decoder; in other words, we must insert in the bitstream a bit that signals the decision about the mode that has been chosed to encode the block.

The `InterImgEncoder` function

This function should encode the image stored in the global variable `img` and create the encoded bitstream as output parameter. In addition, it should use the global variable `decodedFrameBuffer.reference` as a reference for motion estimation and it should store the decoded current image in `decodedFrameBuffer.current`. Here we provide a version for `InterImgEncoder.m` that you have to complete, cut and past into a file, and finally test in the test enviroment provided later on.

```
function [imgBitStream, map]= InterImgEncoder()
% [imgBitStream, map]= InterImgEncoder()
% This function encodes in INTER the image stored in the global variable img
% It uses the global variable params to configure the encoding
% The decoded image is stored in the global variable decodedFrameBuffer.current
% The function returns the encoded bitstream
global params img decodedFrameBuffer MVF

% General initializations
previousDC = 128;
imgBitStream =[];
% Init the cost maps
mapRows = ???;
mapCols = ???;
map.Jintra = zeros(mapRows,mapCols);
map.Jinter = map.Jintra;

% Image pre-processing: not performed in INTER coding, but remember that we
% initialize the DC predictor as 128, which is basically equivalent to the
% pre-processing
```

```
% Loop on the blocks
for rowIndex = 1:params.blockSize:params.nRow,
    for colIndex = 1:params.blockSize:params.nCol,

        % Access to the current block
        currentBlock = img( ???, ???);

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Perform the INTRA encoding of the current block  %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        [blockBitStreamIntra, decodedBlockIntra, quantizedDCIntra] = blockIntraCoding(currentBlock,previousDC);

        % Compute the cost: distortion + lambda * rate
        blockRateIntra = ???; % The number of bits used to encode the block
        blockMSEIntra =  ???; % The MSE of the decoded block with respect to the current block
        codingCostIntra = ???; % J=D+lambda*R, use params.modeSelectionLambda
        % store the coding cost in the output variable map
        % note that in map we have one value per block, so we have to
        % rescale the position onto which we save the value of the cost
        mapRowPos = (rowIndex+params.blockSize-1)/params.blockSize;
        mapColPos = (colIndex+params.blockSize-1)/params.blockSize;
        map.Jintra(mapRowPos,mapColPos) = codingCostIntra;


        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Perform the INTER encoding of the current block  %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        % We need the coordinates of the block inside the blockInterCoding
        % function in order to access to the neighboring blocks and compute
        % the motion vector predictor using the associated vectors
        params.rowIndex = rowIndex;
        params.colIndex = colIndex;

        % The ~ in the output paramenters means that we do not need the
        % third output parameter (the quantized DC). The second input
        % parameter is zero because we do not it neither
        [blockBitStreamInter, decodedBlockInter, ~, motionVector] = blockInterCoding(currentBlock,0);
        % Computing and storing the coding cost
        blockRateInter = ???; % As for the inttra case
        blockMSEInter =  ???; % Idem
        codingCostInter = ???;
        map.Jinter(mapRowPos,mapColPos) = ???;

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Rate-distortion omptimized mode selection (RDO-MS) %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        if  ??? < ???

            % If the condition is met, the INTRA mode is selected

            % Encode a flag for signaling the selected mode and add the encoded block
            imgBitStream =[imgBitStream, '0', ???];

            % Update the decoded frame buffer for the *current* image
            decodedFrameBuffer.current(rowIndex:rowIndex+params.blockSize-1, ...
                colIndex:colIndex+params.blockSize-1) = ???;
           %Update the previous DC
           previousDC = quantizedDCIntra;

           % Even if here the block is encoded in INTRA mode, we need to update
           % the "decoded" motion vector field, in order to perform the
           % right motion vector prediction. In this case, we set the
           % motion vector for the current block to (0,0)
           MVF(rowIndex:rowIndex+params.blockSize-1, ...
               colIndex:colIndex+params.blockSize-1, 1) = 0;
           MVF(rowIndex:rowIndex+params.blockSize-1, ...
               colIndex:colIndex+params.blockSize-1, 2) = 0;

        else
            % The INTER mode has been selected

            % Encode a flag for signaling the selected mode and add the encoded block
            imgBitStream =[imgBitStream, '1', ???];
            decodedFrameBuffer.current(rowIndex:rowIndex+params.blockSize-1, ...
                colIndex:colIndex+params.blockSize-1) = ???;
            % Don't need to update the previous DC

            % Let us update the MVF
            MVF(rowIndex:rowIndex+params.blockSize-1, ...
                colIndex:colIndex+params.blockSize-1, 1) = motionVector(1);
            MVF(rowIndex:rowIndex+params.blockSize-1, ...
                colIndex:colIndex+params.blockSize-1, 2) = motionVector(2);
        end % Mode selection
    end % Loop on Cols
end % Loop on Rows
end
```

Testing the Inter Image coding function

Once you complete the function, you can test it by using the following test environment. Here we encode an image using the Inter coding. The *reference* image used for temporal prediction is the Intra image of a GOP (you can specify the GOP size and the GOP number). For the sake of simplicity, we will use the *original* intra image as reference. You can choose the temporal distance between the reference and the image to be encoded via the `interIndex` parameter.

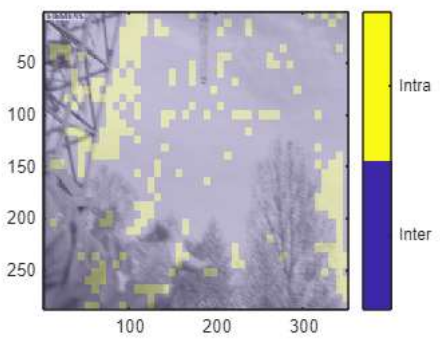
You can also modify the values of λ, λ_{ME} and of the search radius for motion estimation. As for the quality factor, it is defined as `params.qual-params.deltaQ`

If you keep all the parameters unchanged, you should obtain INTER RATE: 0.946 bpp and INTER PSNR: 39.72 dB.

The script also shows the maps of the coding costs for the INTRA and INTER mode, as well as the map of mode decisions.

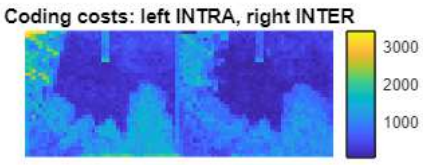
Question 1. Complete the `InterImageCoding.m` function and perform some tests using the provided test environment. Use different video sequences (low motion vs. high motion) and comment about: the RD performance of INTRA coding wrt INTER; the coding cost achievable with the two modes and the relation with the video content (e.g., new objects appearing in the image, areas with low or high details, regular or complicated motion patterns, etc.)

```
clear all; close all;
clc
global params img decodedFrameBuffer MVF

% Setting the configuration: you can change these paramters for testing:
params.inputVideoName = 'foreman'; % Use one of the provided .y files
params.verbose = 0;  % Setting the verboseness level might be useful for debugging and testing
params.qual = 90; % Quality factor for INTRA frame
params.deltaQ = 10; % difference of quality between INTRA and INTER
params.modeSelectionLambda = 10;
params.ME.lambda = sqrt(params.modeSelectionLambda);
params.ME.radius = 8;

params.GOPSize = 5;
gopIndex = 40;       % Choose the GOP index: we will use the first frame as a reference
interIndex = 3;      % this is the "temporal distance" between the reference (INTRA) and the frame to be encoded (INTER)
%-------------------------------------

%-------------------------------------
% Do not change these parameters: -->
params.inputVideoFile   = [params.inputVideoName '_cif.y'];   % full input file name. Add the path if needed
params.blockSize = 8;  %
params.nRow = 288;
params.nCol = 352;
decodedFrameBuffer.reference = zeros(params.nRow,params.nCol); % Since we do not use B, one decoded refernce is enough
decodedFrameBuffer.current   = zeros(params.nRow,params.nCol); % We also need a buffer for the current encoded image
MVF = zeros(params.nRow,params.nCol,2);
params.q_mtx1 =     [16 11 10 16 24 40 51 61;
    12 12 14 19 26 58 60 55;
    14 13 16 24 40 57 69 56;
    14 17 22 29 51 87 80 62;
    18 22 37 56 68 109 103 77;
    24 35 55 64 81 104 113 92;
    49 64 78 87 103 121 120 101;
    72 92 95 98 112 100 103 99];

% define the quantization matrix for Intra frames
SF_Intra = (params.qual==100)+(params.qual<=99&&params.qual>50)*(200-2*params.qual)+ (params.qual<=50)*(5000/params.qual);
params.q_mtx_Intra = ceil(params.q_mtx1*SF_Intra/100);
params.q_mtx_Intra (params.q_mtx_Intra >256)=256;

% define the quantization matrix for Inter frames
params.qual_Inter=max(min(params.qual - params.deltaQ,100),0);
SF_Inter = (params.qual_Inter==100)+(params.qual_Inter<=99&&params.qual_Inter>50)*(200-2*params.qual_Inter)+ (params.qual_Inter<=50)*
params.q_mtx_Inter = ceil(params.q_mtx1*SF_Inter/100);
params.q_mtx_Inter (params.q_mtx_Inter >256)=256;
params.firstFrameGOP =  params.GOPSize*(gopIndex-1) + 1;
% <-- end of parameters that should not be changed
%-------------------------------------


% Let us load two images from the selected GOP
% For the sake of simplicity, we put the original INTRA in the
% decodedFrameBuffer (in real life, here we would have the decoded INTRA
decodedFrameBuffer.reference= loadIntra(gopIndex);
img = loadInter(gopIndex,interIndex);
% Do the encoding!
[imgBitStream, map] = InterImgEncoder();
% Show the image with a color code taht shows the selected mode
mInter = kron(map.Jinter , ones(params.blockSize)); % upsample the maps at image size
mIntra = kron(map.Jintra , ones(params.blockSize));
figure(1); image(ind2rgb(uint8(decodedFrameBuffer.current),colormap("gray")));
hold on;
H1 = image(ind2rgb((mIntra<mInter),colormap(parula(2)))); hc=colorbar;
set(hc,"Ticks", [0.25 0.75], "TickLabels", {'Inter', 'Intra'});
alpha(H1,0.2);
hold off;
```




```
% Show the values of the coding costs
figure(2);
imagesc([mIntra, mInter]); axis image; colorbar;
```

```
imagesc([mIntra, mInter]); axis image; colorbar;
title('Coding costs: left INTRA, right INTER'); axis off
```




```
% Let us compare the RD results with the Intra coding
imgBPPInter = numel(imgBitStream)/numel(img);
PSNRInter = ypsnr(img,decodedFrameBuffer.current);

img = loadInter(gopIndex,interIndex);
imgBitStreamIntra = IntraImgEncoder();
imgBPPIntra = numel(imgBitStreamIntra)/numel(img);
PSNRIntra = ypsnr(img,decodedFrameBuffer.current);
fprintf('INTER RATE: %4.3f bpp\nINTER PSNR: %4.2f dB\nINTRA RATE: %4.3f bpp\nINTRA PSNR: %4.2f dB\n',imgBPPInter,PSNRInter,imgBPPIntr
```

```
INTER RATE: 0.946 bpp
INTER PSNR: 39.72 dB
INTRA RATE: 1.269 bpp
INTRA PSNR: 43.39 dB
```

Complete encoder

Now we can test the full encoder. The provided function will seamlessly integrate with the others, namely the `gopEncoder` and the `sequenceEncoder` functions. **It is adviced to download the latest version of these functions (which is available together with this script) and to overwrite possible previous versions.**

In order to launch the encoding of a video, you have to provide all the parameters and then to call `sequenceEncoder`. In the following section we provide a code that loads a baseline configuration of the encoder:

How to use it: before running a new experiment, first load all the default parameters by launching the following code (press the Load parameters button). Then, go to the section of the experiment you want to run, and launch the experiment, possibly updating the relevant parameters.

```
% Loading the general parameters
clear all; close all;
global params img decodedFrameBuffer MVF

%-----------------------------------
% Setting the configuration

params.inputVideoName = 'bus'; % Use one of the provided .y files
params.verbose = 0;  % Setting the verboseness level is useful for debugging and testing

% Number of encoded gops and gop size
params.numberOfGops = 2; % Start with a small number of GOPS for test, then use the full sequence
params.GOPSize = 1; % GOPSsize=1 means all intra

% Quality control
params.deltaQ = 10; % difference of quality between INTRA and INTER

% RDO parameter
params.modeSelectionLambda = 100;

% ME control
params.ME.lambda = sqrt(params.modeSelectionLambda);
params.ME.radius = 7;

% Compressed file
params.compressedFileName = 'test.bin';
%-----------------------------------


% Do not change these parameters: -->
params.inputVideoFile   = [params.inputVideoName '_cif.y'];
params.blockSize = 8;  %
params.nRow = 288;
params.nCol = 352;
decodedFrameBuffer.reference = zeros(params.nRow,params.nCol);
decodedFrameBuffer.current   = zeros(params.nRow,params.nCol);
MVF = zeros(params.nRow,params.nCol,2);
params.q_mtx1 =     [16 11 10 16 24 40 51 61;
    12 12 14 19 26 58 60 55;
    14 13 16 24 40 57 69 56;
    14 17 22 29 51 87 80 62;
    18 22 37 56 68 109 103 77;
    24 35 55 64 81 104 113 92;
    49 64 78 87 103 121 120 101;
    72 92 95 98 112 100 103 99];
% <-- end of parameters that should not be changed
%-----------------------------------
```

Experiments

Let us perform some experiment: here it is shown a list of possible experiments that you can perform by changing the parameters.

1. Compare different GOP structures: Encode some images of a video sequence using different GOP structures, eg. All Intra, IPIP..., IPPIPP etc. Plot the RD curves. Notice that, if you change the GOP strutcture, you have to chose a different number of GOPs to be encoded in order to have the same number of images in all the experiments. Example: if you compare `gopSize = 2` with `gopSize=8`, in the first case you have to encoder four times as many GOPs than in the second, e.g., 8 GOPs in the first case (in total, 16 images) and two GOPs in the second (again, 16 images)
2. Evaluate the number of Intra blocks in Inter coded frames. Try to understand how this value depends on the video content and on the coding quality factor.
3. Assess the effect of `params.deltaQ`: Keeping unchanged all the other parameters, assess the effect of the different quality used for Intra and Inter frames using the paramater `params.deltaQ` which tells the quality difference between the Intra and the Inter frames of the GOP. Try to understand the relationship with the video content (e.g., static video or dynamic content)
4. Assess the effect of the mode selection lambda. Evaluate the RD curves and the mode selection ratio for different values of lambda. What happens for different sequences?
5. The `sequenceEncoder` function also returns the per-frame PSNR and rate. Draw these curves for the four sequences and try to find the relationship between content, GOP structure and the behavior of PSNR and rate.

You can also design other experiments, in order to understand the impact of the various design choices on the final performance.

Question 2. Perform at least one complete experiment, report the results and comment them. This means:

1. running the experiments for different sequences (note that you don't need to encode the full sequence: a number of images around 20 is typically enough);

2. finding out the relevant parameters (i.e., those that have an impact on the final performance;

3. checking what happens by modifying the parameters / the sequence; for example, change the quality factors, the gop size, the ME parameters...

4. commenting the interaction among the parameters and the results.

Performing Experiment 3 and 4, for which no code is provided, will bring up to one *additional* point to your final grade (up to two points if both are implemented, executed and commented in a complete and correct fashion).

HINT: Consider the the Matlab implementation of the video encoder is relatively slow. It is better to first test the experiment using one single GOP with a small size. Only when sure that everything works correctly, try larger GOP sizes and larger numbers of GOPs.

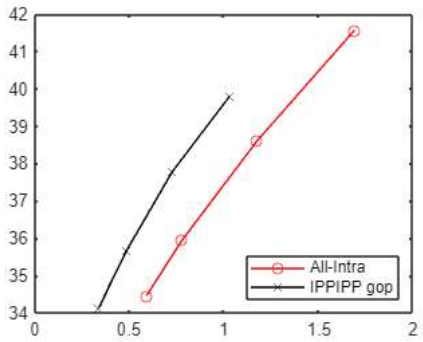
```
%% EXPERIMENT 1
intraQualityFactorVector = [40 60 80 90];
nEncodings = numel(intraQualityFactorVector);
PSNR = zeros(nEncodings, 1);
RATE = zeros(nEncodings, 1);

% ALL INTRA
params.numberOfGops = 3;
params.GOPSize = 1;
for qualityIndex = 1:nEncodings
    params.qual = intraQualityFactorVector(qualityIndex);
    fprintf('Encoding ALL Intra with Q=%d\n', params.qual);
    [R, P] = sequenceEncoder(); %don't need input arg because param is global!
    RATE(qualityIndex) = R;
    PSNR(qualityIndex) = P;
end
```

```
Encoding ALL Intra with Q=40
Encoding ALL Intra with Q=60
Encoding ALL Intra with Q=80
Encoding ALL Intra with Q=90
```

```
% IPP gop
params.numberOfGops = 1;
params.GOPSize = 3; % let us test an IPPIPP... gop structure
PSNR2 = zeros(nEncodings, 1);
RATE2 = zeros(nEncodings, 1);
%% Main loop
for qualityIndex = 1:nEncodings
    params.qual = intraQualityFactorVector(qualityIndex);
    fprintf('Encoding IPP with Q=%d\n', params.qual);
    [R, P] = sequenceEncoder() ;
    RATE2(qualityIndex) = R;
    PSNR2(qualityIndex) = P;
end
```

```
Encoding IPP with Q=40
Encoding IPP with Q=60
Encoding IPP with Q=80
Encoding IPP with Q=90
```

```
% Compare the results
figure; title(params.inputVideoName);
plot(RATE, PSNR,'ro-', RATE2, PSNR2, 'kx-') ;
legend('All-Intra', 'IPPIPP gop', 'Location','southeast');
```




```
drawnow

% Possibly load the parameters before launching this section
```

For this experiment you have to write the code.

```
% Experiment 2 Evaluate the number of Intra blocks in Inter coded frames
% For this experiment you have to modify the functions in order to output
% the map of coding costs, and then use it to compute the number of blocks
% encoded in one or either mode
```

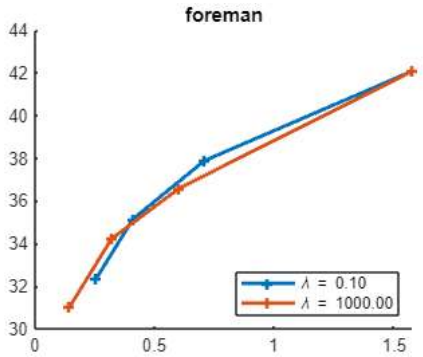
For this experiment you have to write the code.

```
% Experiment 3: Assess the effect of params.deltaQ
% Modify the value of deltaQ and trace the RD curve. Compare different
% choices
```

```
%% Experiment 4  - Effect of lambda

lambdas = [0.1  1000];
intraQualityFactorVector = [ 20 50 80 100];

RMatrix = zeros(numel(intraQualityFactorVector),numel(lambdas));
PSNRMatrix = zeros(numel(intraQualityFactorVector),numel(lambdas));

params.numberOfGops = 2; %
params.GOPSize = 5; %

for lambdaIndex = 1:numel(lambdas)
    for qualityIndex = 1:numel(intraQualityFactorVector)
        params.qual = intraQualityFactorVector(qualityIndex);
        params.modeSelectionLambda = lambdas(lambdaIndex);
        fprintf('Encoding with Q=%d, lambda=%3.2f\n', params.qual, params.modeSelectionLambda);

        [R, P] = sequenceEncoder(); %don't need input arg because param is global!
        RMatrix(qualityIndex,lambdaIndex) = R;
        PSNRMatrix(qualityIndex,lambdaIndex) = P;
    end
end
```

```
Encoding with Q=20, lambda=0.100000
Encoding with Q=50, lambda=0.100000
Encoding with Q=80, lambda=0.100000
Encoding with Q=100, lambda=0.100000
Encoding with Q=20, lambda=1000.000000
Encoding with Q=50, lambda=1000.000000
Encoding with Q=80, lambda=1000.000000
Encoding with Q=100, lambda=1000.000000
```

```
figure; hold on; legendEntries={}; title(params.inputVideoName);
for lambdaIndex = 1:numel(lambdas)
    h=plot(RMatrix(:,lambdaIndex),PSNRMatrix(:,lambdaIndex));
    set(h,'Marker','+', 'LineWidth',2)
    legendEntries{lambdaIndex}= sprintf('\\lambda = %3.2f',lambdas(lambdaIndex ));

end
legend(legendEntries, 'Location','southeast');
```

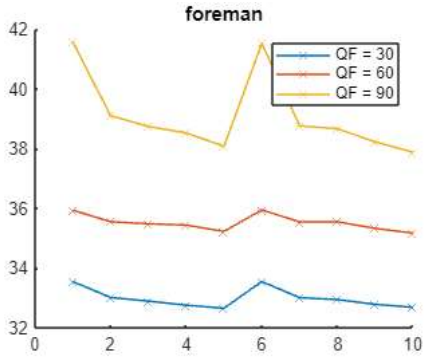



```
% Possibly load the parameters before launching this section
```

```
% Experiment 5
params.GOPSize=5;
params.numberOfGops = 2;

qualities = [ 30 60 90];
figure;
nImages = params.GOPSize*params.numberOfGops;
perFramePSNR = zeros(nImages,numel(qualities));
perFrameRATE = perFramePSNR;

hold on; legendEntries={};

for qIndex = 1:numel(qualities)
    params.qual = qualities(qIndex);
    fprintf('Encoding with Q=%d\n', params.qual);
    [R, P, frameR, frameP] = sequenceEncoder();
    perFrameRATE(:,qIndex)= frameR;
    perFramePSNR(:,qIndex)= frameP;

    legendEntries{qIndex}= sprintf('QF = %d',params.qual);
end
```

```
Encoding with Q=30
Encoding with Q=60
Encoding with Q=90
```

```
%% Draw figures for exp 5

figure; hold on;  title(params.inputVideoName);
for qIndex = 1:numel(qualities)
    plot(1:nImages,perFramePSNR(:,qIndex),'-x');
end
hold off;
legend(legendEntries);
```




```
figure; hold on
for qIndex = 1:numel(qualities)
    plot(1:nImages,perFrameRATE(:,qIndex),'-x');
end
hold off;
legend(legendEntries);
```