Laboratory Exercise 2

Video Compression and Coding using Transforms, Subband Filters and Motion Compensation

Multimedia and Video Communications (SSY150)

Prepared by: Irene Y.H. Gu, irenegu@chalmers.se
Department of Electrical Engineering,
Chalmers University of Technology,
41296, Göteborg, Sweden

Updated in April, 2018

Theoretical Background

A video compression and coding method usually consists of 2 modes: **intra** (or, within-frame) coding mode and **inter** (or, between frame) coding mode. Compression methods used in the intra mode are purely based on compression of each 2D image frame independently. Therefore, compression methods used in intra mode are the same as those of 2D image compression methods. Compression methods used in inter mode is based on image motion compensation among 2 or more adjacent image frames, i.e. only the changing part between adjacent frames are considered. In the following, we briefly review the theoretical background of these two parts.

1 Intra Frame Video Compression based on First Generation 2D Image Compression Techniques

The first generation 2D image compression is usually done in the spatial domain, a transform domain or a subband filtering domain. The compression is achieved by removing small energy signal elements however, without exploiting the perceptually important image properties (e.g. edges, textures of image). Although currently most image compression methods are done using 2nd generation compression techniques or beyond [1, 2], it is important to understand the basics of image compression through learning some basic 1st generation methods. Since image energy is mostly concentrated in the low frequency bands, compression is achieved by removing some small value coefficients in the high frequency bands, typically, by assigning zeros to transformed or filtered coefficients whose values are below a pre-determined threshold. The criterion for choosing a suitable transformation is based on energy compaction (i.e. most energy will be concentrated on a small number of coefficients, while the remaining can thus be removed). It is shown that the DCT (Discrete Cosine Transform) is the 2nd best transform for energy compaction next to the KL (Karhunen-Loeve) transform [3]. The KL transform is an eigenvalue-eigenvector based method and is dependent on the image values), and DCT transform matrix is independent of data.
By removing a certain percentage of small value coefficients in the transform domain (or, in the subband filter domain), and then applying the corresponding inverse transform (or the corresponding synthesis filters), one can obtain an image with a very high compression ratio and reasonably good quality. It is shown that, e.g. a compression ratio of 70-90% can be reached using the DCT, while further compression (or, a better quality image) can be reached by compression in the wavelet filtering domain.

1.1 2D Discrete Cosine Transform (DCT)

Forward 2D DCT: The forward 2D Discrete Cosine Transform for a 2D image $\mathbf{I}(n_1, n_2)$, $0 \le n_1 \le N_1-1$, $0 \le n_2 \le N_2-1$, is described as

$$\mathbf{F}(k_1,k_2) = \frac{c_1 c_2}{\sqrt{N_1 N_2}} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \mathbf{I}(n_1,n_2) \cos\left[\frac{\pi}{2N_1}(2n_1+1)k_1\right] \cos\left[\frac{\pi}{2N_2}(2n_2+1)k_2\right] \tag{1}$$

where $0 \le k_1 \le N_1-1$, $0 \le k_2 \le N_2-1$, further, $c_i = 1$ if $k_i = 0$, otherwise $c_i = \sqrt{2}$, $i = 1, 2$. This results in a frequency domain representation of an image $\mathbf{F}(k_1, k_2)$ by the DCT coefficients $\mathbf{F}(k_1, k_2)$.
As the 2D DCT kernel is separable, the 2D forward transform in (1) can be performed by first applying 1D transforms for individual rows (i.e., $0, 1, \cdots, N_1-1$) followed by 1D transforms for individual columns (alternatively, first column then row transforms), as follows:

$$\begin{aligned}
&\text{Row transforms:} \quad \tilde{\mathbf{F}}(\mathbf{n_1}, \mathbf{k_2}) = \frac{c_2}{\sqrt{N_2}} \sum_{n_2=0}^{N_2-1} \mathbf{I}(n_1,n_2) \cos\left[\frac{\pi}{2N_2}(2n_2+1)k_2\right], \\
&\qquad\qquad 0 \le n_1 \le N_1-1,\ 0 \le k_2 \le N_2-1 \\
&\text{Column transforms:} \quad \mathbf{F}(k_1,k_2) = \frac{c_1}{\sqrt{N_1}} \sum_{n_1=0}^{N_1-1} \tilde{\mathbf{F}}(n_1,k_2) \cos\left[\frac{\pi}{2N_1}(2n_1+1)k_1\right], \\
&\qquad\qquad 0 \le k_1 \le N_1-1,\ 0 \le k_2 \le N_2-1
\end{aligned} \tag{2}$$

Inverse 2D DCT: The inverse DCT for a DCT transformed image $F(k_1, k_2)$, $0 \leq k_1 \leq N_1 - 1$, $0 \leq k_2 \leq N_2 - 1$ is described as

$$\mathbf{I}(n_1, n_2) = \frac{c_1 c_2}{\sqrt{N_1 N_2}} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} \mathbf{F}(k_1, k_2) \cos\left[\frac{\pi}{2N_1}(2n_1+1)k_1\right] \cos\left[\frac{\pi}{2N_2}(2n_2+1)k_2\right] \tag{3}$$

where $0 \leq n_1 \leq N_1 - 1$, $0 \leq n_2 \leq N_2 - 1$, further, $c_i = 1$ if $k_i = 0$, otherwise $c_i = \sqrt{2}$, $i = 1, 2$. This results in an image in the spatial domain that is exactly the same as an original image $\mathbf{I}(n_1, n_2)$.

Similarly, the 2D inverse transform in (3) can be done through inverse transforms of $\mathbf{F}(k_1, k_2)$ along N_2 columns followed by N_1 rows (alternatively, first N_1 rows followed by N_2 columns) as follows:

$$\begin{aligned} &\text{Column transforms:} \quad \tilde{\mathbf{F}}(n_1, k_2) = \tfrac{c_1}{\sqrt{N_1}} \textstyle\sum_{k_1=0}^{N_1-1} \mathbf{F}(k_1, k_2) \cos\left[\tfrac{\pi}{2N_1}(2n_1+1)k_1\right], \\ &\qquad\qquad 0 \leq k_2 \leq N_2 - 1,\ 0 \leq n_1 \leq N_1 - 1 \\ &\text{Row transforms:} \quad \mathbf{I}(n_1, n_2) = \tfrac{c_2}{\sqrt{N_2}} \textstyle\sum_{k_2=0}^{N_2-1} \tilde{\mathbf{F}}(n_1, k_2) \cos\left[\tfrac{\pi}{2N_2}(2n_2+1)k_2\right], \\ &\qquad\qquad 0 \leq n_1 \leq N_1 - 1, 0 \leq n_2 \leq N_2 - 1 \end{aligned} \tag{4}$$

Note that the DCT transform is lossless, i.e., after applying DCT forward and inverse transforms one obtains an exact replica of the original image.

1.1.1 Block-based 2D DCT

For image compression, 2D DCT and 2D inverse DCT are usually applied to small image areas (or, blocks) rather than the entire image area. This is because that most images captured from real world scenes are non-stationary. However, for a small image area, the image can be considered as stationary. Therefore, block-based process is required. This can be analogous to the situation in 1D signal processing: if a 1D signal is non-stationary, we need to divide data into blocks before processing. Data within each block can be considered as stationary when the block size is properly selected. This is exactly the same case for 2D images when we apply block-based DCTs. Block-based DCT is widely used in image compression standards such as JPEG and MPEG. A typical size for the block-based DCTs is 8x8 (or, 16x16) pixels [5, 6].

1.2 2D Subband Filters

Subband filters can be considered as a generalized signal transformation. An orthogonal transformation of a signal is equivalent to decomposing signal into multiple bands, where the number bands is equal to the total length of the transformation matrix. For example, applying a 1D DCT of length 2 to a signal results in 2 bands. Applying a 2D DCT transformation of length 2x2 is equivalent to 2 subband filters in each direction (x or y) , followed by downsampling by 2 in each direction, resulting in 4 subband images, i.e., L_xL_y, L_xH_y, H_xL_y, H_xH_y bands. Here, L is lowpass, H is highpass, and the subscript x or y indicates x or y direction. Subband filters also decompose a signal into multiple bands. However, they differ from the transformation in that: (a) filter kernels (i.e. the filter impulse responses) do not need to have the same length as the number of bands; (b) filter kernels for different bands do not need to have the same length. Furthermore, subband filters have many advantages over a block-based transformation, for example, the processed image does not have the so-called block artifact as compared with the block-based approach where block artifact is common. Also, one can design subband filters based on the very mature filter theories. It is worth noting that in the subband filter theory, analysis filters and synthesis filters should be designed jointly such that a signal passes the analysis filters followed by the synthesis filters satisfies the condition of *perfect reconstruction* (i.e., there is no distortion or aliasing errors to the original signal, except some time delays and a constant scaling factor to the signal magnitude). Often, other conditions are posed, such as orthogonality, linear phase and FIR filters. Theories and details on designing a perfect reconstruction filter bank under these constraints are beyond the scope of this laboratory exercise. Some relevant materials can be found in [7, 8, 9].

1.2.1 Subband filters for 1D signals

Let the original 1D signal be $s(n)$, the impulse responses of 1D analysis filters be $h_i(n)$, and the impulse responses of 1D synthesis filters be $g_i(n)$, $i = 1, \cdots, M$. To decompose a 1D Signal into frequency bands, $i = 1, \cdots, M$, applying analysis filters to the original signal gives:

$$s_i(n) = s(n) * h_i(n) = \sum_k s(k) h_i(n-k), \quad i = 1, \cdots, K \tag{5}$$

where $h_i(n), i = 1, \cdots, K$ are the impulse responses of K subband filters $H_i(z),\ i = 1, \cdots, K$, and $H_0(z)$ is the lowpass filter, $H_1(z), \cdots, H_{K-1}(z)$ are the bandpass filters, and $H_K(z)$ is the highpass filter. This set of filters will cover the entire signal bandwidth. These filters may have the same bandwidth, or different bandwidth (see the 4th plot in the left hand side of Figure 3, where the subband filters have different bandwidths related by an octave bandwidth ratio). Designing the analysis filters and synthesis filters are usually done jointly, in order to achieve the requirement of *perfect reconstruction*, and with other possible constraints (e.g. linear phase, orthogonality). As the filtered signal after $H_i(z)$ has a reduced bandwidth, $s_i(n)$ can be down-sampled. A maximum down-sampling rate of M could be applied to $s_i(n)$.

The reconstruction of a 1D signal is done by using the synthesis filters. First, the subband signal is upsampled (if downsampling has been applied after the analysis filters). After that, the synthesis filters can be applied followed by summing up over all bands $i = 1, \cdots, M$. Assuming no downsampling, the following reconstructed signal is obtained:

$$\hat{s}(n) = \sum_{i=1}^{M} s_i(n) * g_i(n) = \sum_{i=1}^{M} \sum_k s_i(k) g_i(n-k) \tag{6}$$

If no signal processing is done in the subband filtering domain, then the reconstructed signal $\hat{s}(n)$ in (6) shall be the exact replica of the original signal $s(n)$, except that a constant delay and a constant scaling are allowed, $\hat{s}(n) = cs(n - n_0)$.

Perfect reconstruction (PR): The design of analysis filters and the synthesis filters are done jointly, such that has the PR condition is satisfied. The PR condition requires the aliasing error between different analysis filters and synthesis filters be zero. Furthermore, a signal that passes an analysis filter bank followed by a synthesis filter bank will only be allowed to have a constant scale in magnitude and a constant time delay of n_0. The PR condition can be summarized by:

$$\begin{array}{l} \sum_{i \neq j} H_i(z) G_j(z) = 0 \\ \sum_i H_i(z) G_i(z) = c z^{-n_0} \end{array} \tag{7}$$

where c, n_0 are constants.

Compression and filtering in the frequency domain: Often, some processing of the signal is desirable in the frequency domain, e.g. compression by removing small value frequency coefficients, or denoising by removing some high frequency components. Signal reconstruction is then followed after the processing using the synthesis filters.

1.2.2 Subband filters for 2D signals

When applying 2D subband filters to images, we will relate 1D and 2D subband filters in a similar way that we have previously related the 1D and 2D DCTs. We shall assume that 2D filter kernels are separable (even though this may not be the case for subband filters!), so that we can apply 1D filters along the rows, and subsequently along the columns. Of course, we may first apply the filters along the columns then followed by rows. Noting that assuming a separable kernel is merely for simplifying the processing. If the true filter kernels are nonseparable, then assuming separable will result in an image filtered by less desirable filters.

1.3 Subband Filters using Wavelets

Subband filters may be implemented by using wavelet transforms, e.g. Multiresolution analysis (MRA) of an image using dyadic wavelets filters. Wavelet filters are related to special types of subband filters that satisfy certain conditions such as *admissibility* (implying that the Fourier transform of $\psi(t)$ vanishes at the zero frequency: $|\psi(t)|_{\omega=0} = 0$), and *regularity* (implying that the wavelet function $\psi(t)$ should have some smoothness and concentration in both time and frequency domains). Regularity is a concept linked with *vanishing moments* (or the approximate order of wavelets): if $M_p = \int t^p \psi(t) dt = 0$ for $p = 1, \cdots, n$, then the wavelet function has n vanishing moments. Roughly speaking, regularity and vanishing moments indicate how fast a wavelet decays. There are many advantages of wavelet filters: high energy compaction (e.g. higher than the DCTs), multiresolution decomposition of a signal (i.e. filters with octave bandwidths, see Fig.3), finite support (or, finite kernel length) of filter kernels, and many more. Most details are beyond the scope of this laboratory work. For those who are interested in more details, you are referred to some further reading [10, 11].

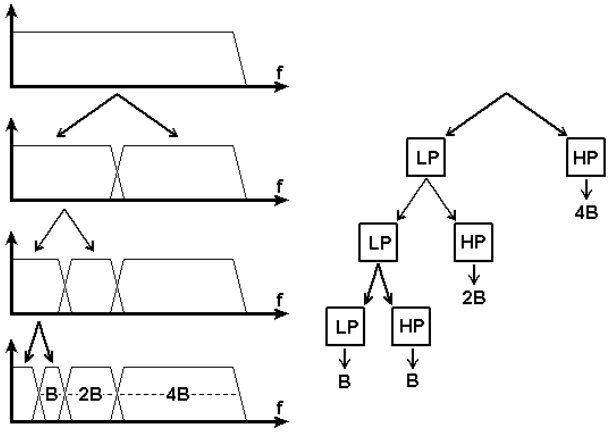

Figure 1: Subband 1D signal decomposition by using wavelets with 3 levels: subbands and their bandwidth relations. *Left*: from top to bottom: the original signal bandwidth (8B bandwidth); 1st level of wavelets splits the signal into 2 bands (4B bandwidth each); 2nd level of wavelets splits the lowpass band into 2 bands (2B bandwidth each); 3rd level of wavelets splits the 2nd level lowpass band into 2 bands (B bandwidth each). This results in a total of 4 bands (equals to the total number of levels + 1). *Right*: an equivalent description of the subbands generated by multiscale wavelets (in the left figure) by using a unbalanced binary tree.

Wavelets for Image Compression: Wavelets have been widely used for image compression. The wavelet domain is known to offer better image compression than that does in the DCT domain. Wavelets are included in the image compression standard JPEG 2000.

2 Inter-Frame Video Compression based on Block-Matching and Motion Compensation

This section describes methods used for the inter coding mode. In video coding standards, inter coding mode is applied to P and B pictures (P stands for prediction frames and B bidirectional prediction frames).

A good match during the search means that a good prediction can be made from the block in the previous image frame to obtain the block in the current frame. Once the "best" matching is found for a motion block, the corresponding motion vector and the prediction error are computed. The (encoded) motion vector (dx, dy) and some large prediction error values are then sent. A good matching requires

that the whole block has undergone the same translation, and the block should not overlap objects in the image that have different degrees of motion, including the background. A simplest and typical way to do video compression is to use a technique commonly referred to as "block-matching". The technique was originally described by Jain and Jain [14]. It is rather easy to implement, and hence widely adopted. In the method, each image frame is first divided into a fixed number of rectangular (usually, square) blocks, followed by searching the "best" matching in the reference frame (usually the next frame of image).

Area of search: For each block in the frame, a search is made in the reference frame. The search can either be performed over the full image range (i.e., *global search*), or, constrained within a maximum allowed displacement area (i.e., *local search*).

Uni-directional and bi-directional block matching
Block matching can be performed in one direction, or two directions. For each motion block, e.g. (i, j)-th block ((i, j) is the spatial position on the top-left corner of the block) in the current image frame $\mathbf{I}_t(x, y)$, block matching MC can be performed by

$$\mathbf{I}_t(x_i, y_j) = \sum_k w_k \mathbf{I}_{t-k}(x_i - dx(k), y_j - dy(k)) \tag{8}$$

where $\sum_k w_k = 1.0$, $k = t - L, \cdots, t - 1, t + 1, \cdots, t + L$ for bi-directional MC, and $k = t - L, \cdots, t - 1$ for one directional MC, searching of displacement vectors $(dx(k), dy(k))$ is performed in the entire image $\mathbf{I}_{t-k}$ area for the global search.

Criterion for selecting the best matching block: The aim of the search is to find the "best" matching block, which gives the least prediction error. Commonly used criteria are based on some kind of average matching errors. The two most frequently used criteria are the *mean square error (MSE)* criterion, and the *mean absolute error (MAE)* criterion that is easier to compute. The best matching block is then found for the image frame $\mathbf{I}_t(x, y)$ such that the MSE/MAE value is minimized.

Suppose that we choose the MSE criterion. Then, for the given current image frame $\mathbf{I}_t(x, y)$ at the time instant t and the (i, j)-th block ($i, j = 1, \cdots,$), the block matching is to find the best prediction block in the previous image frame $\mathbf{I}_{t-1}(x, y)$ by selecting the displacement vector (dx, dy) such that the MSE of the block matching is minimized among all possible displacement vectors $(dx, dy) \in \mathcal{R}_I$, where $\mathcal{R}_I$ is the search area.

More specifically, this can be done in the following steps:

(a.1) Compute the MSE for the (i, j)-th block (i is the block index along the x direction, and j is along the y direction) with a candidate displacement (dx, dy) as follows:

$$\begin{aligned} MSE_{(i,j)}(dx, dy) = \; & \tfrac{1}{M_x M_y} \sum_{y_j=(j-1)M_y+1}^{jM_y} \sum_{x_i=(i-1)M_x+1}^{iM_x} [\mathbf{I}_t(x_i, y_j) - \mathbf{I}_{t-1}(x_i - dx, y_j - dy)]^2 \\ & 1 \le (x_i - dx) \le N_x, \; 1 \le (y_j - dy) \le N_y \text{ (for global search)} \end{aligned} \tag{9}$$

where $M_x \times M_y$ is the block size along the x and y direction, respectively, and $N_x \times N_y$ is the image size. The summation is over the pixels within the (i, j)-th block, i.e., $[x_i, y_j] \in [(i-1)M_x + 1 : iM_x; \;\; (j-1)M_y + 1 : jM_y]$. For a global search, the range of displacement vector (dx, dy) is within the entire image area, i.e., $(x_i - dx; \; y_j - dy) = (1 : N_x; \; 1 : N_y)$. The computation is repeated over all possible combinations of (dx, dy) values. Further, one should store and update the smallest $MSE(dx, dy)$ and the corresponding displacement vector (dx, dy) during the search process, or, alternatively, to store all the MSE values.

(a.2) Then, the displacement vector that results in the smallest MSE is selected as the "motion vector",

$$(dx^*, dy^*) = \text{argmin}_{(dx,dy)} MSE_{(i,j)}(dx, dy) \tag{10}$$

The starting pixel position (i.e. the pixel in the up-left corner) of the matched block is then $(x_i - dx^*, y_j - dy^*) = ((i-1)M_x + 1 - dx^*, (j-1)M_y + 1 - dy^*)$ in the previous image frame $\mathbf{I}_{t-1}(x, y)$.

In (9), the search is performed over the whole image area, and the solution is the *global optimal solution*. However, if the search is performed over a constrained area (i.e. local search), e.g. $dx \in \{-dx_{max/2}, \cdots, dx_{max/2}\}, dy \in \{-dy_{max/2}, \cdots, dy_{max/2}\}$, then the search area in (9) should be changed correspondingly, and the solution from using (9) and (10) is a *local optimal solution*. For applying a local search, a typical example value for the maximum displacement is 64 pixels from the original position of a block. However, this value is mainly dependent on the relative motion speed under the given frame rate that captures the video.

Alternatively, if one chooses to apply the MAE criterion, the block matching is done by:

(b.1) Replacing (9) by the following equation to compute the MAE for the ith block,

$$MAE_{(i,j)}(dx, dy) = \frac{1}{M_x M_y} \sum_{y_j=(j-1)M_y+1}^{jM_y} \sum_{x_i=(i-1)M_x+1}^{iM_x} |\mathbf{I}_t(x_i, y_j) - \mathbf{I}_{t-1}(x_i - dx, y_j - dy)| \\ 1 \le (x_i - dx) \le N_x,\ 1 \le (y_j - dy) \le N_y \text{ (for global search)} \tag{11}$$

(b.2) Replacing (10) by using following equation to choose the best displacement vector for the prediction block in the current image frame,

$$(dx^*, dy^*) = \text{argmin}_{(dx,dy)} MAE_{(i,j)}(dx, dy) \tag{12}$$

Block size: A typical block size is on the order of $M_x \times M_y$ = 16x16 pixels and is fixed. Although a *fixed block size* is most often utilized, more sophisticated methods may used a *variable block size*. The choice of block-size used for motion compensation is always a compromise. Smaller blocks can better represent the complex motion than fewer large ones, however, it require more computations and more motion vectors. Larger blocks may result in less motion vectors thereby less transmission rate, however, there might be more large prediction errors required to be encoded. Overall, selecting the block size is a compromise between the transmission rate (both motion vectors and encoded prediction errors), computational complexity, block artifact, and the image quality.

Searching strategy: Although one may apply some strategies for fast searching (for reducing the computational cost), the most straightforward and simple way is to apply an exhaustive search. This could be computationally demanding in terms of data throughput, but algorithmically it is simple and relatively easy for hardware implementation.

3 Example of a Video CODEC System

3.1 Video Encoding

The block diagram in Figure 2 briefly sketches the MPEG-4 video encoding system. In a video coding system, coding is split and switched between two modes: the *inter* and the *intra* mode. Details on the standardized coding systems such as MPEG-4 or H263L, H264/AVC are beyond the scope of this laboratory. There exist rich information which can easily be found in video coding books [15, 16, 17] or in presentations through Internet search.

INTRA mode: The intra coding mode only uses the spatial information in an individual image frame for the coding, and hence it can be considered as a 2D image encoding approach. Intra frame coding is usually applied to the 1st image frame, and for the "refreshment" frames (referred to as "I pictures") in a regular frame interval. The aim of regularly applying INTRA mode coding is to prevent the propagation

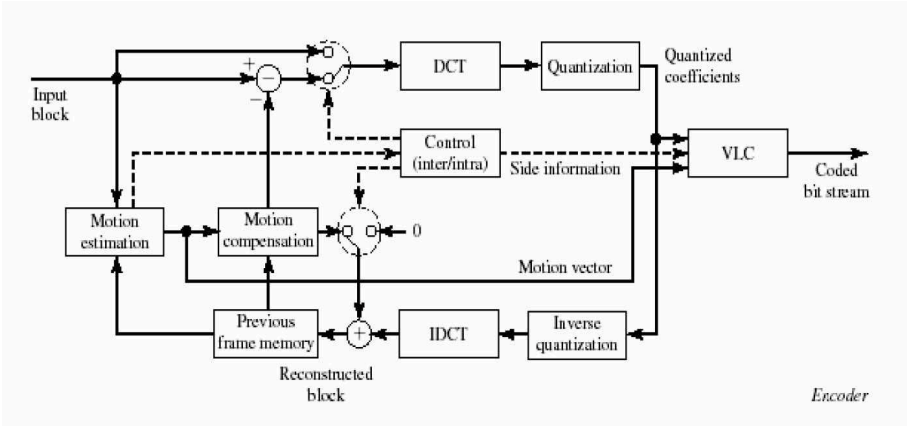

Figure 2: Using motion compensation for video encoding. (Courtesy of Dr.Yao Wang [18])

of coding errors through image frames. Since the coding methods applied to the INTRA mode are exactly the same as in the 2D image compression, we will not include this coding part (see Laboratory-2 in this course for details).

INTER mode: The inter coding mode is associated with motion prediction and compensation along the temporal direction. As sketched in the block diagram of Fig.2, the steps in the INTER coding mode can be described as follows:

(a) Motion estimation:
In the block "motion estimation", one can apply, e.g. block matching by searching blocks from the previous image frames (which have undergone the process of DCT transform, quantization, inverse DCT transform and inverse quantization). The output from this block is a motion vector (dx, dy), which is sent to the VLC (variable length encoder) that forms the bit steam. The output is also fed back to the "motion compensation" block.

(b) Motion compensation:
In the motion compensation block, the predicted block is obtained by assigning the shifted block from the previous frame $\hat{\mathbf{I}}_t(x_i, y_i) = \mathbf{I}_{t-1}(x_i - dx, y_i - dy)$ (see Equations (9)-(12)). If the motion vector is zero, then one just needs to copy the corresponding block from the previous image frame.

(c) Encoding of prediction errors:
The prediction errors from the motion compensated block are then computed by $e_i(x_i, y_i) = \mathbf{I}_t(x_i, y_i) - \hat{\mathbf{I}}_{t-1}(x_i - dx, y_i - dy)$. After that, the prediction errors are DCT transformed and the small value DCT coefficients are removed. The remaining values are quantized before sending to the VLC (variable length coding), which generates binary codes and forming the coded bit scream of the encoder. The output is now ready for transmission on an IP network.

3.2 Video Decoding

The video decoder is an inverse process of the video encoder, as shown in the block diagram of Figure 3. This includes inverse VLC to obtain the quantized DCT coefficients and the motion vectors. The quantized DCT coefficients are then inverse quantized and inverse DCT transformed to form the prediction errors. While motion compensated blocks are formed by using the given motion vectors which direct to the corresponding blocks from the previous frame. The block obtained from the motion compensation and the prediction errors of that block are then added to obtain the decoded block (or, reconstructed block) in the current frame. For simplifying this laboratory exercise, the steps of encoding and decoding the prediction errors from block matching will be neglected.

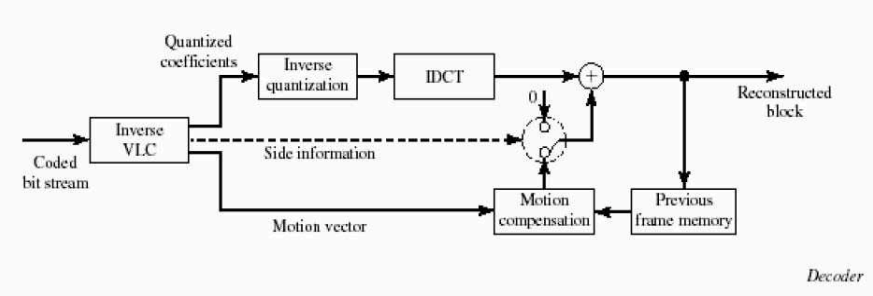

Figure 3: Decoding video by using the coded bit stream of video obtained from the receiver. (Courtesy of Dr.Yao Wang)

4 Assessing the Quality of Compressed Video/Images: Objective Quality Measures

Measuring image quality is a complex task, and is highly perceptually related - depending on the human visual systems. Detailed description of subjective and objective criteria for image quality is beyond the scope here. For those who are interested in the subject, some further readings can be found in [4] and the references therein.
Here, we shall only describe two of the most commonly used objective criteria: peak signal-to-noise ratio (PSNR) and structure similarity (SSIM).

4.1 PSNR measure

It is worth mentioning that PSNR is far from a complete description of the image quality. However, by combining PSNR with the visual observation one can somewhat determine the image quality. Usually, images with PSNR values higher than 30dB are considered as acceptable quality. The PSNR (peak signal-to-noise ratio) between the original image $\mathbf{x}$ and the reconstructed image $\mathbf{y}$ is defined as:

$$PSNR = 10 \cdot \log_{10}\left(\frac{\mathrm{MAX}_{\mathbf{x}}^2}{\mathrm{MSE}}\right) \tag{13}$$

where the mean squared error is $\mathrm{MSE} = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\left(\mathbf{x}(i,j) - \mathbf{y}(i,j)\right)^2$, $\mathrm{Max}_{\mathbf{x}}$ is the dynamic range of the image (e.g. for an image $\mathbf{x}$ with $B = 8$ bits/pixel **unsigned char**, $\mathrm{Max}_{\mathbf{x}} = 2^B - 1 = 255$).

4.2 SSIM measure

The structural similarity (SSIM) metric measures the similarity between two images, for example, the original image and the uncompressed image. The SSIM measure is usually considered better than PSNR since it takes into the image structure into account, however, computing SSIM is more complicated than that of PSNR. SSIM is designed to improve on traditional methods like peak signal-to-noise ratio (PSNR) or mean squared error (MSE) which are known to be less consistent with human visual system perception. The SSIM measure is defined on the *local* images $\mathbf{x}$ and $\mathbf{y}$ as

$$SSIM(\mathbf{x},\mathbf{y}) = \frac{(2\mu_{\mathbf{x}}\mu_{\mathbf{y}} + c_1)(2\sigma_{\mathbf{xy}} + c_2)}{(\mu_{\mathbf{x}}^2 + \mu_{\mathbf{y}}^2 + c_1)(\sigma_{\mathbf{x}}^2 + \sigma_{\mathbf{y}}^2 + c_2)} \tag{14}$$

where $\mu_{\mathbf{x}}$ is the mean value of $\mathbf{x}$, $\mu_{\mathbf{y}}$ the mean value of $\mathbf{y}$, $\sigma_{\mathbf{x}}^2$ is the variance of $\mathbf{x}$, $\sigma_{\mathbf{y}}^2$ the variance of $\mathbf{y}$, $\sigma_{\mathbf{xy}}$ the covariance of $\mathbf{x}$ and $\mathbf{y}$, $c_1 = (k_1 L)^2$ (or, $c_2 = (k_2 L)^2$ is a small constant that avoids the instability when the denominator $(\mu_{\mathbf{x}}^2 + \mu_{\mathbf{y}}^2)$ (or, $(\sigma_{\mathbf{x}}^2 + \sigma_{\mathbf{y}}^2)$) is very close to zero, L is the dynamic range of the pixel values (for an image with B-bits/pixel, $L = 2^B - 1$), $k_1 = 0.01$ and $k_2 = 0.03$ by default. Practically, SSIM is computed on each pixel with a local window (e.g. 8×8) between the two given images $\mathbf{x}$ and $\mathbf{y}$,

and then the MSSIM (Mean SSIM) is computed over all pixels in the images. This can be described as follows:

1. Partitioning two images $\mathbf{x}$ and $\mathbf{y}$ into small and non-overlapped blocks (e.g, for simplicity, this is equivalent to applying a rectangular 8x8 window. However, other window function can be applied as well, for example, Gaussian shape window);

2. For each corresponding block in images $\mathbf{x}_{i,j}$, $\mathbf{y}_{i,j}$ (where (i, j) is the top left corner coordinates of an image block, $i = 1, \cdots, M$, $j = 1, \cdots, N$), compute the local $\text{SSIM}(\mathbf{x}_{i,j}, \mathbf{y}_{i,j})$ within the local image block using (14).

3. Repeat the above step 2 for blocks in $\mathbf{x}_{i,j}$ and $\mathbf{y}_{i,j}$.

4. Compute the Mean SSIM (MSSIM) by taking the average SSIM values as follows:

$$\text{MSSIM}(\mathbf{x}, \mathbf{y}) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \text{SSIM}(\mathbf{x}_{i,j}, \mathbf{y}_{i,j}) \tag{15}$$

To further improve the results (avoiding block artifacts), [12] replaces the rectangular window by a Gaussian window (e.g., size 11×11 with a standard deviation of $\sigma = 1.5$), so that the pixels near the boundaries of window are less weighted. More details can be found in [12, 13].

References

[1] M. Kunt, A. Ikonomopoulos and M. Kocher, "Second generation image coding techniques", Proceedings of the IEEE 73(4):549-575, 1985.

[2] M.M.Reid. R.J.Millar, N.D. Black, "2nd-generation image coding: an overview", ACM computing surveys (CSUR), pp.3-29. vol.29, No.1, 1997.

[3] William K. Pratt, Digital Image Processing, 4th ed., John Wiley & Sons Inc., 2007.

[4] H.Wu, K.R.Rao and K. Ramamohan Rao, Digital Video Image Quality and Perceptual Coding, CRC Press, 2005.

[5] M. Ghanbari, Standard Codecs: Image Compression to Advanced Video Coding, IEE, 2003.

[6] Roger J.Clarke, Digital Compression of Still Images and Video, Academic Press, 1995.

[7] Sanjit K. Mitra, Digital Signal Processing: a computer-based approach., 3rd ed., McGraw-Hill, 2006.

[8] L.R. Rabiner, Digital Processing of speech signals, Prentice-Hall Inc.

[9] Ali N. Akansu and Richard A. Haddad, Multiresolution Signal Decomposition: Transforms, Subbands and Wavelets, Academic Press, 2nd ed., 2001.

[10] Gilbert Strang and Truong Nguyen, Wavelets and Filter Banks, Wellesley -Cambridge Press,1996.

[11] User's Guide for Matlab Wavelet toolbox, The MathWorks Inc.

[12] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity", IEEE Trans on Image Processing, vol. 13, no. 4, pp. 600-612, 2004.

[13] https://ece.uwaterloo.ca/˜ z70wang/research/ssim/ (last visit: 2011-03-07)

[14] J.R. Jain and A.K. Jain, "Displacement measurement and its application in interframe image coding", IEEE Trans. Commun., Vol. COM-29, No. 12, pp. 1799-1808, Dec. 1981.

[15] Abdul H. Sadka, Compressed video communications, John Wiley & Sons, Ltd, 2002.

[16] Iain Richardson and Iain E. G. Richardson, H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia, John Wiley & Sons, Ltd, 2003

[17] Markus Flierl and Bernd Girod, Video Coding with Superimposed Motion-Compensated Signals: Applications to H.264 and Beyond (The International Series in Engineering and Computer Science), Kluwer Academic Publisher, 2004.

[18] http://eeweb.poly.edu/∼yao

5 Laboratory Exercises

This laboratory exercise is aimed at hands-on learning of video compression techniquesby:

(a) INTRA video coding using 2 of the first-generation 2D image compression methods: DCT transform and wavelets;

(b) INTER video compression based on block matching motion compensation.

For INTRA coding mode, DCT and wavelets will be applied separately on a single video frame (or, one image). For INTER coding mode, block matching motion compensation will be applied on 2 video frames. You will learn these through making your own Matlab programs and doing simulations.

Summary of the Tasks

For INTRA-mode, the tasks are summarized as follows:

1.1 : *2D DCT (Discrete Cosine Transform) for image compression.*
- Apply 2D DCT to the original image.
- Compress DCT coefficients by setting zeros to small value coefficients.
- Apply inverse 2D DCT, resulting in the compressed image.
- Compute the PSNR (Peak Signal-to-Noise Ratio) and Mean SSIM (mean structural similarity) for the compressed image.

1.2 : *Block-based 2D DCT for image compression (for nonstationary images*).
- Partition the image into non-overlapped 8×8 blocks of equal size.
- For each block, apply the 2D DCT, then compress DCT coefficients (i.e., set small values to zeros), and finally apply the inverse DCT to obtain the reconstructed image block after compression.
- Repeat the above process over all blocks.
- Compute the PSNR (Peak Signal-to-Noise Ratio) and Mean SSIM (Mean structural similarity) for the compressed image.

1.3 : *2D subband filters (discrete wavelet transform) for image compression.*
- Apply 2D "analysis wavelet filters" by using Discrete Wavelet Transform from the Matlab wavelet toolbox, resulting in a set of filtered sub-images (choose the number of scale levels =4, which is equal to 5 bands for a 1D signal).
- Compress the image by choosing a small threshold and setting filtered image values below the threshold to zeros.
- Apply 2D "synthesis wavelet filters" to the thresholded subband images, resulting in a compressed image.
- Compute the PSNR (Peak Signal-to-Noise Ratio) and Mean SSIM (mean structural similarity) for the compressed image, and compute the compression ratio.

For INTER-mode, the sub-tasks are summarized as follows:

2.1: Learn how to read avi video files, play *movie*, display and save image frames, and convert image formats in Matlab.

2.2: Compute motion blocks by thresholding the difference image.

2.3: Estimate motion vectors for motion blocks using the MAE criterion and the global search.

2.4: Apply motion compensation in motion blocks. For no-motion blocks, copy the corresponding image blocks from the previous frame.

2.5: Objective quality measures using PSNR (Peak Signal-to-Noise Ratio) and MSSIM (Mean structural similarity) for the reconstructed image.

Exercises:

A: INTRA-Frame Image Compression

Task 1.1: DCT-based Image Compression

Step (1.1):
First, download the image sequence (in the "avi" format) from the course website, and then save the video file. Read the *.avi video from Matlab, and display the video. Extract images by selecting 2 image frames (e.g. 10 frames apart) that clearly contain object motion, and save the selected image frames as *.bmp images. If the video is colored, then convert the image frames to gray-scale ones, and then save them as *.bmp images.

Some useful Matlab functions:
video=VideoReader('filename.avi'): read avi video into Matlab;
movie(video): play the movie with the file name 'video'
length(video): total number of frames in the video:
I=frame2im(video(j)): extract the jth frame from video to image I
size(I): determine the size of I, i.e., number of rows, columns, color or gray image
rgb2gray: convert an rgb image to a gray-scale image.
imwrite(I,filename,'bmp'): e.g. filename='frame5', then image named as 'frame5.bmp' will be saved to the disk. Other useful Matlab functions: imread, imshow).

Step (1.2) Apply 2D DCT to an entire image (use Matlab function: dct2), resulting in the image containing DCT coefficients $\mathbf{F}(n_1, k_2)$ in the frequency domain. Plot the 2D DCT transformed image $\mathbf{F}(k_1, k_2)$.
(**Important**: Before applying the DCT, make sure that the image format is properly converted, where the range of pixel values is between 0.0 and 1.0. (Useful Matlab functions: mat2gray (convert format).

Step (1.3): Image compression: Removing small value DCT coefficients
Select the compression ratio as 90% and select the corresponding threshold value th for $\mathbf{F}(k_1, k_2)$ (for simplicity, a global threshold th is applied). Replace with zero values to these DCT coefficients whose absolute values are below the threshold th. This results in a compressed image $F_1(k_1, k_2)$ in the frequency domain.

Hint: sort the absolute values of DCT coefficients in an ascending order, pick up the coefficient value at the position $th = (0.90 * N_1 * N_2)$, and assign it as the threshold. Set zeros to those DCT coefficients that satisfy $|\mathbf{F}(k_1, k_2)| \leq th$.

Step (1.4): Obtain a compressed image by inverse DCT.
Apply the inverse 2D DCT to $F_1(k_1, k_2)$, resulting in a compressed image in the spatial domain $\mathbf{I}_1(n_1, n_2)$. Compute the error image $\mathbf{e}_1(n_1, n_2) = |\mathbf{I}(n_1, n_2) - \mathbf{I}_1(n_1, n_2)|$. Compute the PSNR for the image after compression using (13) and Mean SSIM using (15).

Results from task 1.1. to be included in the report:

- Threshold value $th =$ used, resulting PSNR and Mean-SSIM for the compressed image.
- Plot 4 images: the original image $\mathbf{I}(n_1, n_2)$, the image in the DCT domain $\mathbf{F}(k_1, k_2)$, the compressed image $\mathbf{I}_1(n_1, n_2)$ after inverse DCT, and 30 times enlarged error image $30{*}\mathbf{e}_1(n_1, n_2)$.
- Comment the results from visual observation of these images.

Task 1.2: Block-based DCT-domain Image Compression

Step (2.1): Read and display a 2D image in gray scale, by repeating Step (1.1).

Step (2.2): Divide the image into non-overlapped blocks, each is of size 8x8 (pixels). Apply the 2D DCT to each block of image, and insert the DCT transformed block into the corresponding location. Repeat this process over all blocks. Finally, this results in a DCT block transformed image $\mathbf{F}_2(k_1, k_2)$.
View the DCT block transformed image $\mathbf{F}_2(k_1, k_2)$.

Step (2.3) Compression in the DCT domain: Choose a fixed (global) threshold value (which would set approximately 90% of the DCT coefficients to zeros), and set zeros to DCT coefficients if their absolute values are below this threshold.

Step (2.4) Inverse block DCT transform: For each block (the same size as the forward DCT: 8x8), directly apply the inverse 2D DCT (Matlab function: idct2), and insert the resulting block of data into the corresponding image location. Repeat this process for all blocks. Finally, this results in the image after the compression $\mathbf{I}_2(n_1, n_2)$.

Step (2.5): Compute the error image $\mathbf{e}_2(n_1, n_2) = |\mathbf{I}(n_1, n_2) - \mathbf{I}_2(n_1, n_2)|$, and compute the corresponding PSNR_2 and MSSIM_2 according to (13) and (15).

Results from task 1.2 to be included in the report:

- The threshold value th used; the corresponding PSNR PSNR_2 and mean SSIM value MSSIM_2.
- Plot 4 images: the original image $I(n_1, n_2)$, the block transformed DCT image $\mathbf{F}_2(k_1, k_2)$, the compressed image $I_2(n_1, n_2)$ in the spatial domain, and the error image (enlarge 30 times) $30 * e_2(n_1, n_2)$.
- Compare the results from applying the DCT to the entire image, and from applying block-based DCTs, and write down your comments.

Task 1.3: Subband Image Compression using Wavelets

This exercise will be done by using the existing GUI software under the Matlab wavelet toolbox, named 'wavemenu'.

Step (3.1): Type 'wavemenu' to start the GUI of the wavelet toolbox.

Step (3.2) Load the original image (use the same image as in Step (1.1.1)): Click the box 'Wavelet 2-D', which activates a new window. Under the new window, choose 'File', then 'Load image' to specify an existing image.

Step (3.3) Apply wavelet analysis filters for image decomposition: Select a wavelet "bior" (biorthogonal wavelets), choose '6.8' (which are related to the lengths of filter kernels) for the analysis and synthesis kernels, and chose 'level=4' (4 scale level meaning 5 bands for 1D, or 13 bands for 2D signals). Click 'Analysis' for image analysis: this results in 3 relatively large sub-images in the bands (LH, HL and HH), and the next level which is a further decomposition of LL into 4 smaller sub-images (LL, LH, HL, HH). This process is repeated for each level. You can view each band of sub-image by clicking 'Visualize'.

Step (3.4) Compression in the wavelet domain and saving the compressed image: Clicking 'Compress' will activate another window, where you can select a threshold, e.g., by 'Select Global threshold': you may observe that the yellow line moves in the lower plot, also the corresponding value of 'Number of zeros' in percentage (which is the compression ratio). Select the threshold value as the desired compression ratio (select the ratio as 90%). Finally, click 'Compress'. Observe the result and save the compressed image as $\mathbf{I}_3(n_1, n_2)$.
(hint: select *"File"* $\rightarrow$ *"save"* $\rightarrow$ *"compressed image"* from the GUI).

Step (3.5) Compute the error image $e_3(n_1, n_2) = |I(n_1, n_2) - I_3(n_1, n_2)|$. Then, compute $PSNR_{Wavelets}$ according to (13) and $MSSIM_{wavelet}$ according to (15) for the wavelet compressed image, and compare with the corresponding $PSNR_{BlockDCT}$ and $MSSIM_{BlockDCT}$ values from the block-based DCT compressed image (using the same compression ratio!). Plot the wavelet compressed image and the block-based DCT compressed image (with the same compression ratio!) and compare by visual observation of the results from these two methods.

Results from task 1.3 to be included in the report:

- The actual compression ratio used (choose to be the same as the block-based DCT!). The PSNR values from wavelets and from block-based DCT compressed image (i.e., $PSNR_{Wavelets}$ and $PSNR_{BlockDCT}$), and Mean SSIM values from wavelets and from block-based DCT compressed image ($MSSIM_{Wavelets}$ and $MSSIM_{BlockDCT}$).
- Plot 4 images: the original image $\mathbf{I}(n_1, n_2)$, the wavelet compressed image $\mathbf{I}_3(n_1, n_2)$, the error image (enlarge 30 times) $30 * e_3(n_1, n_2)$, and the block DCT compressed image $\mathbf{I}_2(n_1, n_2)$.
- Write down your comments after comparing the PSNR, MSSIM and visual quality of the compressed images from wavelets and block-based DCT.

B: INTER-Frame Motion Compensation and Compression

Task 2.1: Read and extract image frames from video

Learn how to read avi video files, play *movie*, display and save image frames, and convert image formats in Matlab by repeating Step (1.1).

Task 2.2: Compute motion blocks by thresholding the difference image

Load 2 consecutive image frames that were previously saved. Assigned these 2 images with the name I_{old} and I_{new}, respectively. Compute the difference image from the two image frames ($I_{diff} = I_{new} - I_{old}$). Apply a fixed threshold to the difference image (i.e. Initially set a threshold value, e.g. th=50/255 (for image value ranges [0,1]). Set zeros to those pixels whose absolute values are below the threshold (i.e. if $|I_{diff}(i, j)| < th$, then $I_{diff}(i, j) = 0.0$).

Task 2.3: Estimate motion vectors for motion blocks using MAE criterion and global search

Divide the image area into rectangular blocks (size 16×16). For each block indexed as (j, i): if there is one pixel within the block of I_{diff} contains the motion, then assign this block as a motion block, otherwise, it is a non-motion block. Repeat this process over all blocks. Generate a map image (same size as the original image) for indicating motion blocks and name it as I_{motion}: for those pixels within the motion blocks, assign image values as 1.0, otherwise 0.0.

Some useful Matlab functions:
imread: read an image from a file

mat2gray: convert an image, from the pixel value range: [Imin,Imax] to a matrix image with a range [0.0 1.0].
Note: **converting between image formats** is very important for Matlab image processing. When usng *mat2gray*, the image pixels extracted from video are usually 8 bits unsigned char (range between 0-255), while most Matlab processing requires pixels in float/double format. There are many other functions for image format conversion, e.g. ind2gray, gray2ind, rgb2gray. You can use Matlab's help to find out more. Display 3 images: the 2 (original) consecutive images, and the motion image "Imotion". Observe the motion blocks, and change the threshold value th until the motion blocks indeed include all prominent changes in the image frames. Plot these 3 images, and give the final threshold value.

Plot the resulting I_{motion} image, and *observe* whether or not the motion blocks cover most motion areas in the original image: if not, then change the threshold th value until I_{motion} shows satisfactory motion areas.

Results from tasks 2.2 and 2.3 to be included in the report:

- Finally selected threshold value th
- Plot 4 images: I_{new}, I_{old}, I_{diff}, I_{motion}.

Task 2.4: Motion compensation for motion blocks

Compute motion vectors: For each motion block in the current image frame I_{new}, use a full search to find the "best" block under the MAE (mean absolute error) criterion (see Eq.(11) in the previous image frame I_{old}. Note, searching a best matching block in the previous image should be applied in a highest spatial resolution (i.e. each time, only one pixel shift is applied in each step for block matching). Pick up the best motion vector for each block in the current image frame, and save the motion vectors into a matrix MV =MV $(1 : No\_blocks\_in\_y\_direction,\ 1 : No\_blocks\_in\_x\_direction,\ 1{:}2)$, where MV contains the corresponding best motion vector (dx, dy) for each block (specified by its top left coordinates (j, i)). Repeat this process to all motion blocks indicated by I_{motion}.

Motion compensation for motion blocks: First, initialize the motion compensated image as zeros, $I_4 = zeros(sisey, sizex)$. For each motion block specified by the index (j, i), its image values in I_4 are replaced by the values in the shifted block in I_{old}, where the displacement is specified by the motion vector. That is, for the block with the index (j, i) in the current image I_{new}, the top-left corner pixel index for the block $(yold, xold)$ in the previous image frame I_{old} is $[(j-1)*\text{blocksize}+1-dy, (i-1)*\text{blocksize}+1-dx]$. Repeat this process for all motion blocks.

INTRA processing of non-motion blocks by copying previous blocks: First, copy the motion compensated image I_4 to a new image I_5. For those non-motion blocks (i.e. $I_{motion}(block) = 0$, copy the image values in the block $I_{old}(block)$ to the corresponding block in $I_5(block)$. This results in these blocks of INTRA processed non-motion blocks. Finally, compute the error image $e_5 = |I_{new} - I_5|$.

(Note, for simplifying this laboratory work, the errors from motion compensated blocks are not encoded. Further, for simplicity, the steps "quantization" and "inverse quantization", "VLC encoding" and "inverse VLC decoding" in Figures 2 and 3 are excluded in this laboratory exercise).

Task 2.5: Compute objective image quality by using the PSNR and MSSIM

Compute the PSNR and MSSIM for the reconstructed image I_5, where the original image I_{new} is used as the ideal image.

Results from tasks 2.4 and 2.5 to be included in the report:

- Plot 4 images: I_{new} (the original image), I_4 (motion compensated image from INTER mode), I_5 (reconstructed image from both INTER and INTRA modes), error image (enlarge 30 times) $30 * e_5$. item resulted PSNR(I_5) and MSSIM(I_5) values.

Task 3: Writing a Report

Write a report including: your test results, your Matlab programs, summary and comments to INTRA and INTER video compression techniques learned from this laboratory exercise. In addition, the following discussions should also be included in your report:

1. Briefly describe how 2D image compression is achieved, and how video compression is achieved. (no more than 4 lines of text!)
2. To achieve high image compression, what is the best 'criterion' you select a transform ? (brief, no more than 2 lines of text!)
3. Wavelet vs. DCT for 2D image compression: compare and interpret the results: (a) do you expect (intuitively) that the wavelet would generate better quality images? (b) Did your experiments agree with your intuition ? (c) If your intuition does not agree with your test results, try some different test settings: increase the level of wavelet transform also increase the compression ratio, and then compare the results. Do you observe the image quality changes from the wavelet vs. the DCT ? (d) Can you draw a preliminary conclusion on wavelet vs. DCT in terms of 2D image compression?
4. What is the main property that is exploited in INTER video compression for achieving high compression rate ? (brief, no more than 2 lines of text!)
5. Suppose motion compensation (MC) in video is done by using a block matching technique through **local** searching, as both computational cost and reconstructed MC video quality are important. Assuming 95% of the motion vectors should be best/optimally compensated, how would you determine the best local searching area size in the block matching ?
6. For applying motion compensation, what are main differences between a local search and a global search approach in terms of quality and computation. (brief, no more than 3 lines of text!)