

Real-Time Digital Video Watermarking

Matthew Harper and Harsh Juneja, TAMU
Course Instructor: Professor Deepa Kundur

Objective

This lab introduces students to the basic Digital Watermarking techniques applied to video-processing software using Simulink and its hardware implementation using the TI-6437 EVM signal processing board.

Introduction and Background

Advances in digital image processing and digital video processing have opened incredible possibilities for video streaming applications [1]. Moreover, as broadband Internet becomes more prevalently available, and multimedia resources become increasingly accessed, protecting intellectual property and copy rights of multimedia products is not getting any easier [2]. However, digital watermarking mechanisms can help ensure the integrity of copyrights. As a result, the watermarking process has increasingly gained interest within the video and audio fields. Currently, the primary watermarking transformation algorithms include the spread spectrum, the DCT transformation and DWT the transform method. This lab introduces the student to the Discrete Cosine Transform (DCT) algorithm and its application to digital watermarking. Once the watermark is embedded into the video data stream, further video processing can include compression, decompression, and extraction.

The DCT Algorithm

Although there are several transformations utilized in digital video processing, the DCT continues to be one of the most common linear transformations within digital signal processing [3]. By performing lossy compression, the DCT separates the image into parts of differing importance. The 2D-DCT and 2D-IDFT displayed below in figures 1 and 2 respectively, can not only concentrate the main information of original image into the smallest low frequency coefficient which enables a reduction in overall computational complexity. That is, by concentration the majority of the frequencies in the initial coefficients, we can reduce computation by performing processing solely on those elements [2]. Moreover, this process performs robustly when applied to compression and decompression storage and retrieval techniques.

$$F(jk) = a(j)a(k) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(mn) \cos\left[\frac{(2m+1)j\pi}{2N}\right] \cos\left[\frac{(2n+1)k\pi}{2N}\right]$$

$$f(mn) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} a(j)a(k)F(jk)\cos\left[\frac{(2m+1)j\pi}{2N}\right]\cos\left[\frac{(2n+1)k\pi}{2N}\right]$$

Constructing the Model

As we have observed in previous labs and lectures, there are many video processing applications aided by the use of the DCT. Because of the complexity of these real-time systems it is often useful to build the model in a series of stages. Breaking up a complex image/video processing tasks into smaller more specialized stages allows one to more easily replace a component with a more advanced algorithm [1].

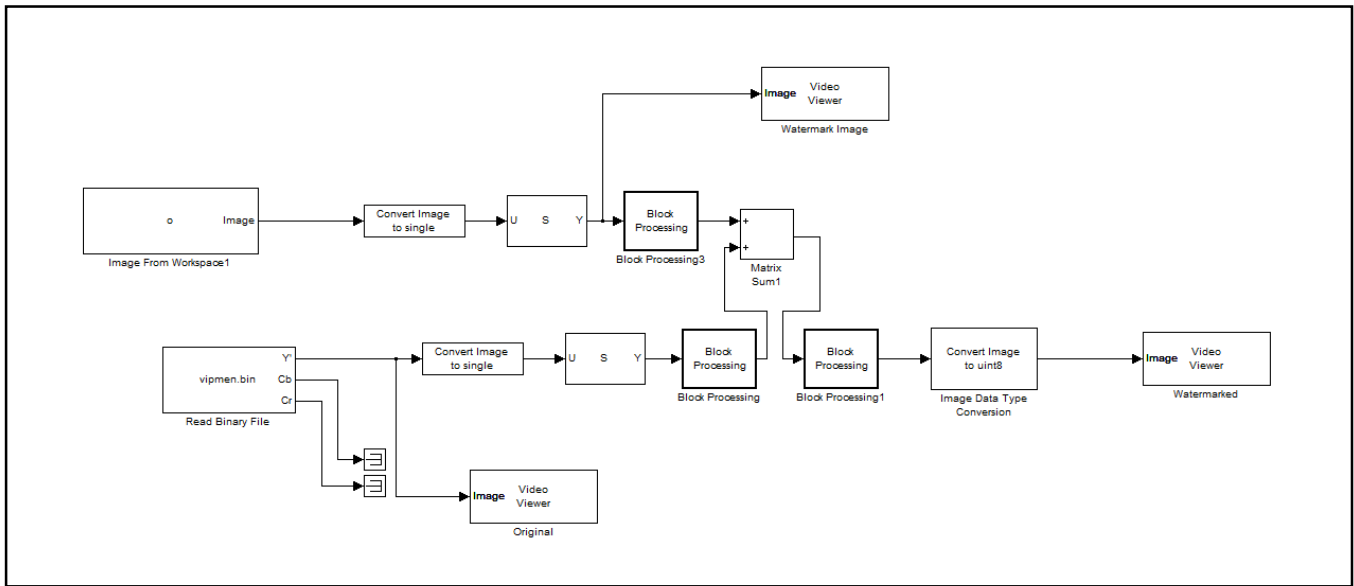


Figure 1: Simulink Digital Watermarking Model.

Design and Implementation

In this stage, a Simulink model is developed for a software implementation of video watermarking on a video file.

Building the Simulink Model

1. Insert the “Read Binary File” block from the Video and Image Processing Blockset. This will enable us read a binary video file stored on the computer. Within the settings for this block, browse for the file “vipmen.bin” which should be the video to be watermarked and select it.
2. Insert an “Image Data Type Conversion” block and connect its input to the output Y’ port of the “Read Binary File” block from the previous step. The output data type parameter for the “Image Data Type Conversion” block should be set to single. Also, terminate the output ports Cb and Cr of the “Read Binary File” using “Terminator” blocks. The output port Y’ of the “Read Binary

File” should also be connected to a “Video Viewer” block so that the original video can be viewed simultaneously along with the watermarked video.

3. Next, add a “Selector” block and connect its input to the output port of the “Image Data Type Conversion” block. The selector block enables a specified number of elements in a multi-dimensional input signal to be processed and ensures proper size. For the “Selector” block, change the number of input dimensions to 2 and the index mode to zero-based. For the first dimension, set index option to starting index (dialog), index to 0, and output size to 112. For the second dimension, set index option to starting index (dialog), index to 0, and output size to 160. The sample time should be set to -1. It is usually beneficial to enable the signal dimensions under Format - port/signal displays.
4. The output of the “Selector” block should now connect to a “Block Processing” block. The block extracts sub matrices of a user-specified size from the input matrix. For DCT operations this parameter should be set to powers of 2 (ex. {[8 8]}). It then sends each sub matrix to a subsystem for processing, and then reassembles each subsystem output into the output matrix. The parameters number of inputs, number of outputs, overlap, and traverse order should be set to 1, 1, {[0 0]}, and row-wise respectively. Click on the “Open Subsystem” button and within it add a “2-D DCT” block with default values between the inport and the output.
5. Now, add an “Image From Workspace” block and connect its output to the input of a new “Image Data Type Conversion” block with the output data type parameter again set to single. The “Image from Workspace” block reads the image that will be watermarked onto the digital video. This can be accomplished by saving an image to the set path in the workspace and using the imread command (`h=imread('text.tif');`).
6. Repeat step (3) after step (5) with the same parameter values.
7. Connect the output of the “Selector” block from the previous step to a “Video Viewer” block so that the user can see the image that will be watermarked onto the video. This output should also connect to another “Block Processing” block and the exact same procedure in step (4) should be followed with the exact same values.
8. Connect the outputs of the “Block Processing” blocks from the previous step and from step (4) to a “netsum” block. The parameters for list of signs and sample time to ++ and -1 respectively.
9. The output port of the above “netsum” block should feed into a new “Block Processing” block. The parameters for this block should again match those values listed in step (4). However, the subsystem changes. Click on the “Open Subsystem” button and within it add a “2-D IDCT” block with default values between the inport and the output.

10. The output port of the above “Block Processing” block should feed into an “Image Data Type Conversion” block with the output data type set to uint8.
11. The output of the above “Image Data Type Conversion” block is now the required watermarked video that is being sought. Connect this output to a “Video Viewer” block to view the results.

Digital Video Hardware Implementation

1. Go to the Simulation→Configuration Parameters, change the solver options type back to “Fixedstep”, and the solver to “discrete (no continuous states).”
2. Start with your Simulink Model for digital video edge detection and delete all the input and output blocks but leave the “Image from Workspace” block.
3. Under Target Support Package→Supported Processors→TI C6000→Board Support→DM6437 EVM, found Video Capture block. Set Sample Time to -1.
4. Add two Video Display blocks. For the first one, set Video Window to “Video 0”, Video Window Position to [180, 0, 360, 480], Horizontal Zoom to 2x, Vertical Zoom to 2x; for the Second one, Set Video Window to “Video 1”, Video Window Position to [0, 240, 360, 240], Horizontal Zoom to 2x, Vertical Zoom to 2x.
5. Add a Deinterleave block, and two Interleave blocks from DM6437 EVM Board Support. Link the Video Capture block to the Deinterleave block. And Link the two Interleave blocks each with a Video Display.
6. For computational efficiency output video stream components Y, Cb, and Cr can be resized to a lower sample size for processing. This can be accomplished by inserting a resize block for each video component after the initial Deinterleave block. The lower resize value should always be a factor of 32 if the default Interleave mask value of 32 is utilized. This can be calculated by first dividing the initial video mask size by a factor of two and rounded to the nearest number of a factor of 32. Then by dividing by 2 once more. For example our Y video component for the gray scale was originally 720. $(720 \times 480) / 2 = 360 \times 240$. However, 352x256 is the nearest factor of 32. Then by dividing by 2 we obtain the entered value of 176x128. This process should be repeated for Cb, and Cr to get 88x128.
7. From the Simulink library add a DIP block for the DM6437EVM and connect it to a multi-port switch. The DIP switch setting should be set to SW4(0) and sample time of 1. Connect the output pin of the DIP to the top pin of the multi-port switch. Double click on the multi-port switch and set the number of input port to 2 (++). Connect one of the inputs to the output of the Image from Workspace block. This will allow us to simulate the insertion of the watermark image when unauthorized access is detected.
8. Press Ctrl + B to begin upload and execute the program.

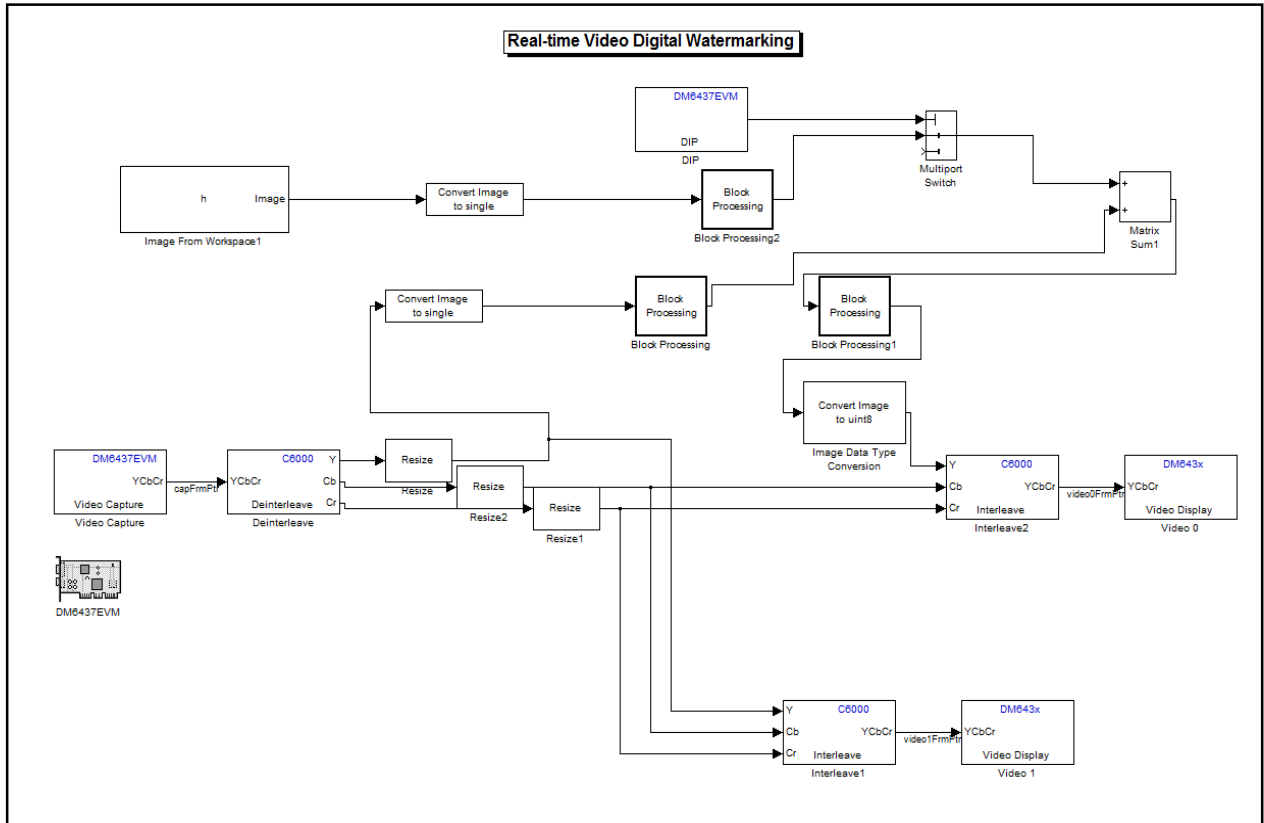


Figure 2: Real-Time Digital Watermarking model

Results



Figure 3: Original, Watermarked, and Watermark Masking Images respectively.

References

- [1] Kundur, Dr. Deepa, Texas A&M University, "ECEN 448 Real-Time Digital Signal Processing." Last modified DEC 11, 2011. Accessed December 13, 2011.
<http://www.ece.tamu.edu/~deepa/ecen448/>.
- [2] Kougianos, Elias, Saraju P. Mohanty, and Dhiraj K. Pradhan. University of North Texas., "Simulink Based Architecture Prototyping of Compressed Domain MPEG-4 Watermarking." Last modified DEC 01, 2011. Accessed December 01, 2011.
http://www.cse.unt.edu/~smohanty/Publications_Conferences/2009/MohantyICIT2009Watermarking.pdf.
- [3] LaJianshengst, FiMei, Li Sukang, and Tan Xiaomei. Academy Publishers, "A Digital Watermarking Algorithm Based On." Last modified DEC 01, 2011. Accessed December 13, 2011.