

# Lab 3: Digital Watermarks

## Individual Project

Due June 1, 11:59PM via Turnin (use lab3)

### Part I. Implement the spread spectrum watermark embedding [60pt] (estimated time: 1--2 hours)

- Read original image (get lena.jpg included in the lab3 package), make it 8-bit grayscale
  - Via matlab: **rgb2gray()**, **uint8()**
- Generate watermark vector  $w$  of length  $n$  (e.g.,  $n = 1000$ )
  - Via matlab: **randn()**
  - **Note: try to make  $w$  as zero mean so it can be negative or positive**
- Apply 2D DCT transform on the **entire image, not each macroblock**
  - Via matlab: **dct2()**
- Locate the  **$n$  largest** AC coefficients  $x$ 
  - Via matlab: **sort()**
- Generate watermarked coefficients  $x'$  by  $x' = x * (1 + a * w)$ 
  - $w$  is the corresponding watermark component,  $a = 0.1$
- •Apply 2D IDCT on the new DCT coefficients (original DC, new  $x'$  and the rest AC coefficients, make sure to put them in original order)
  - Via matlab: **idct2()**
- Compare the original and watermarked image
  - Compute the MSE & PSNR
- Repeat the above with different  $n$  (100, 200, 500, 1000, 1500)
- Plot a figure of PSNR vs.  $n$ , report your finding in a word document
- Your matlab function: **[psnr,w]=ss\_embed(srcfilename, wmkfilename, n)** , it outputs the **PSNR** of the watermarked image, the watermark **w**, and writes the watermarked image to wmkfilename using jpeg no compression.

## Part II. Implement the spread spectrum watermark detection

[40pt] Detect/extract watermark  $w$  (estimated time, 1--2 hours)

- Detect whether a watermark  $w$  is present in a test image, use the  $w$  in Part I, read the test image from the testfilename specified below.
- 
- Apply 2D DCT on the image
  - Via matlab: `IDCT2()`
- Extract the  $n$  largest coefficients
  - (Hint: use the original image to identify the location of these  $n$  coefficients)
- Subtract the corresponding  $n$  DCT values of the original image, let the vector be  $y$
- Compute the similarity between  $w$  and  $y$ , use a threshold of 6 to check whether  $w$  is present

$$\text{sim}(Y, W) = \frac{\langle Y, W \rangle}{\sqrt{\langle Y, Y \rangle}} \quad \langle Y, W \rangle = \sum_i y_i \cdot w_i$$

- Run this function using the original unwatermarked image and the watermarked image, report your findings on the similarity, and its dependency on  $n$
- Submit your code, and report your findings in a separate document
- Your matlab function is :  
**`sim=ss_detection(srcfilename,testfilename,w,n)`**

where `srcfilename` is the original image, `wmkfilename` is the test image,  $w$  is the watermark to be tested, and  $n$  is the watermark embed length.

**Bonus: Examining the robustness of your watermarks**  
**[25pt]**

- Add noise to your watermarked image, see if you can detect the original watermark
  - Via matlab: `noisyimg = imnoise(wimg,'gaussian');`
- Compress & decompress your watermarked image (JPEG compression with quality 25 and 50), and see if you can detect the original watermark
- Produce collusion attacks
  - Generate three different watermarked ( $w_1, w_2, w_3$ ) images following the same manner, take the average of the three to create a new image
  - Detect whether any watermark ( $w_1, w_2, w_3$ ) is present
- Evaluate the above with different values of  $n$  specified in Part I, report your findings on the similarity values