## Contents

# 1. Lab 1

## *1.1. Exercise 1*

The aim of the first exercise was to successfully import to MATLAB and convert to Y'CbCr scale the provided image.

- Load the provided RGB image - *lena512color.tiff*.
  ```
  im = imread('lena512color.tiff');
  ```

- Display the image as well as its separate RGB channels in grayscale (Fig. 1) .
  ```
  imR = im(:,:,1); imG = im(:,:,2); imB = im(:,:,3);
  ```

- Convert the image in RGB scale into Y'CbCr scale (Fig. 2) solely exploiting matrix operations:

$$V = \left(AU^T + [0\ 128\ 128]^T\right)^T \tag{1}$$

  where $A$ is a given transformation matrix, $U^T$ is a transpose of the RGB image matrix and $V$ is the the Y'CbCr image matrix. Both matrices must be reshaped from $(N, N, 3)$ to $(N^2, 3)$ for the above formula to be applicable.


**Fig 1:** Entire *Lena* image displayed as well as its separate RGB channels (in grayscale).


**Fig 2:** The *Lena* image converted to the *Y'CbCr* scale and displayed in individual channels (in grayscale).

### 1.2. Exercise 2

To goal of the second exercise of this lab depended on finding the source device of an image based on its subsampling method.

- Load the provided *Y'CbCr* image - *kimono1_1920x1080_150_10bit_420.raw*. Due to the image being encoded with 10-bit precision, the file ought to be cast to at least a 16-bit precision matrix. To ensure correct import, the number of records must be specified, which in a 4:2:0 format, is equivalent to an entire 1920x1080 $Y$ bit plane and two 960x540 *Chroma* planes.

```
fid = fopen('kimono1_1920x1080_150_10bit_420.raw','r');
im = fread(fid, 1920*1080*1.5, 'uint16');
fclose(fid);
```

- The file stores the *Y'CbCr* channel information in a sequential order, hence, can be decomposed and reshaped in to appropriate dimensions as follows (Fig. 3):

```
dims = 1920 * 1080;
Y = im(1 : dims); Y = reshape(Y, [r c]);
Cb = im(dims+1 : dims+(dims/4)); Cb = reshape(Cb, [r/2 c/2]);
Cr = im(dims+(dims/4)+1 : dims+(dims/2)); Cr = reshape(Cr, [r/2 c/2]);
```

- Subsample the $Y$ plane by compressing every 2x2 $Y$ pixel block into a single pixel value in each of the four possible *modes*:
  - **A**: average of all four pixel intensity values,
  - **B**: average of the two left-most pixel intensity values,
  - **C**: average of the two right-most pixel intensity values,
  - **D**: a single the top-left pixel intensity value

- Zero-pad the $Y$ plane and selected *Chroma* plane (here *Cb*), otherwise the planes are indivisible into 16x16 blocks.

- Adopt a block-based approach and compare every subsampled Y block with the Chroma component using the *square of correlation coefficient* (Fig. 4).

- Select the mode with the largest number of blocks with highest correlation factor and the corresponding source device. The results for this image are:

```
suspectID=D; nblocks=3290;
```

‘


**Fig 3:** The loaded RAW 4:2:0 format video frame displayed in individual *Y'CbCr* channels (in grayscale).

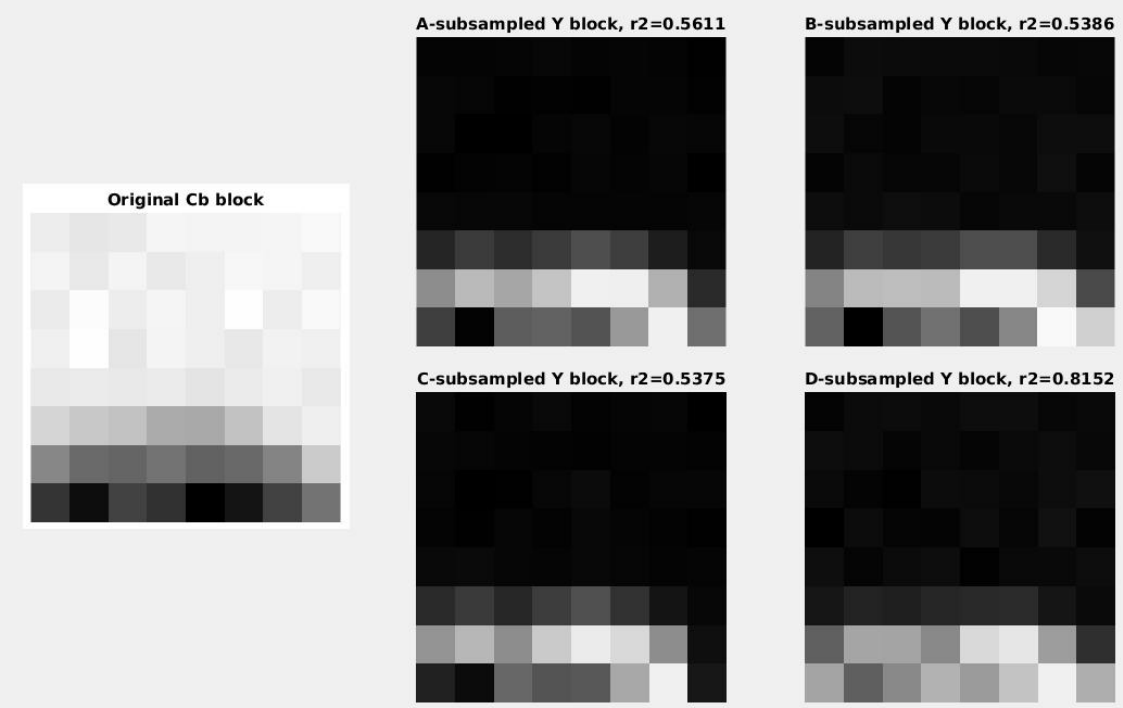


**Fig 4:** A selected block of the original *Chroma* (*Cb*) and corresponding subsampled *Y* blocks in four specified modes. Each subsampled block has its calculated *r2* similarity to the Chroma block appended.

## 2. Lab 2

### *2.1. Exercise 1*

The first exercise of this lab concerned the image enhancement in the pixel domain obtained through histogram matching.

- Import the *color_cast.png* image (Fig. 5), separate into RGB channels and calculate corresponding normalised histograms (Fig. 6).
- Import the *hist_ref.png* grayscale image and display its normalised histogram (Fig. 7).
- Map the CDF's (*cumulative distribution function*) of above histograms to the CDF of the reference image (*hist_ref.png*). The results are three images of enhanced channels (Fig. 8).
- Combine the modified channels into a single enhanced RBG image (Fig. 9).

**Fig 5:** Original *color_cast.png* image with visible red colour cast.

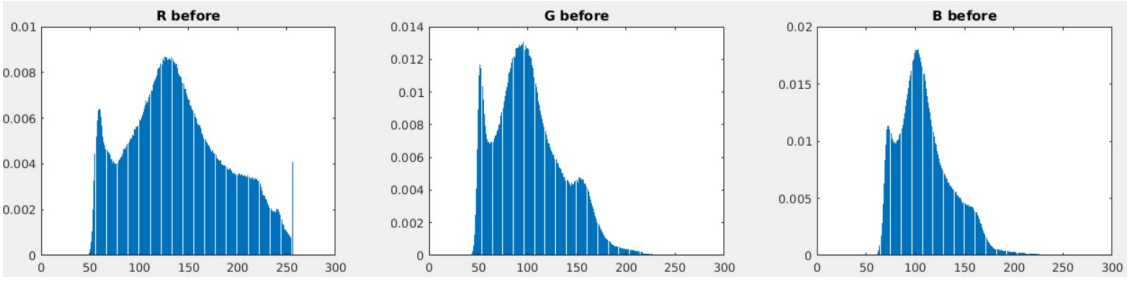


**Fig 6:** Normalised histograms of the RGB channels of the original image.

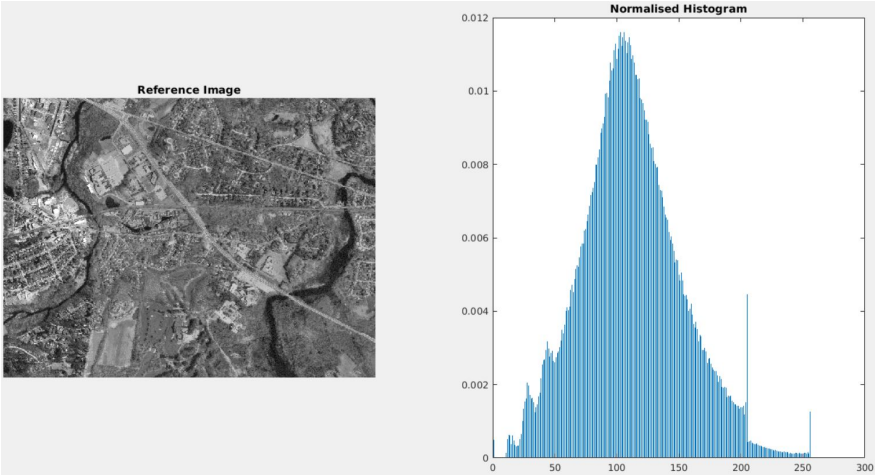


**Fig 7:** The grayscale reference image (*hist_ref.png*) and its corresponding normalised histogram.

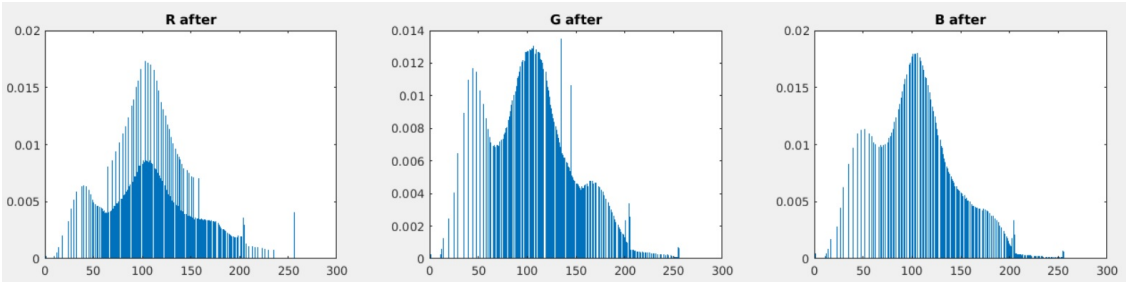


**Fig 8:** Normalised histograms of the enhanced RGB channels obtained by histogram matching.


**Fig 9:** Entire enhanced image consisting of its histogram-matched channels.

### 2.2. Exercise 2

The task of the latter exercise was based on repetitive spectral noise removal.

- Import the noisy image (*halftone_evidence.pgm*), calibrate padding parameters employing the given `paddedsize()` method and convert it into the frequency domain using the discrete Fourier transform (Fig. 10).

- Manually locate the bright peaks (in the unshifted DFT) and remove them using the given `notch` filter function which utilises the Butterworth high-pass filter:
  ```
  H1 = notch('btw', PQ(1), PQ(2), 40, x, y, 2);
  ```
  where `PQ(1); PQ(2);` are dimensions of the image after padding and `(x,y)` are the coordinates of the frequency peaks in the DFT. It is important to exclude the `(0,0)` peak as it corresponds to the low frequencies which removal leads to high information loss.

- Combine the individual notch filters into a single spectrum filter through bit-wise multiplication. Apply the inverse discrete Fourier transform (IDFT) to convert the image back to the pixel domain. (Fig. 11).

- To further denoise the image, remove the horizontal and vertical lines by setting their value to zero in the frequency domain. Convert the image to spectral domain using IDFT (Fig. 12).

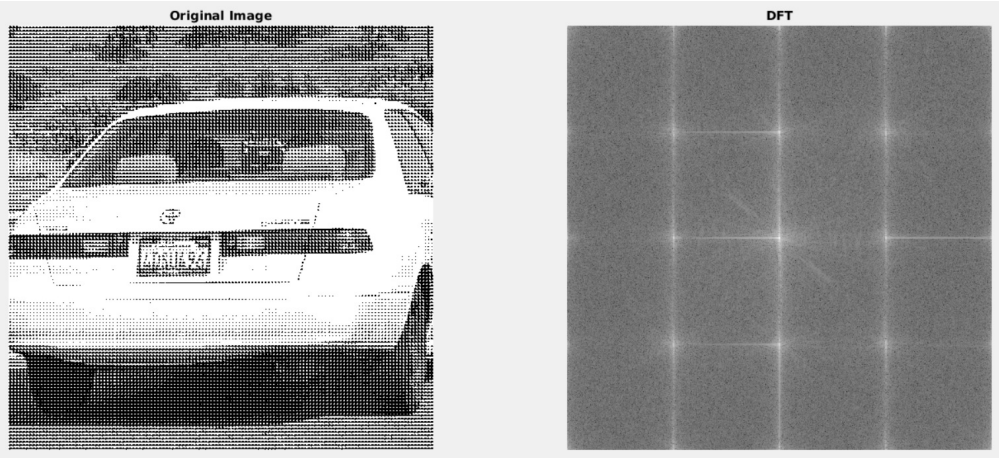


**Fig 10:** Original image with repetitive noise and the resulting shifted discrete Fourier transform.


**Fig 11:** Denoised image displayed in the frequency and spectral domain.

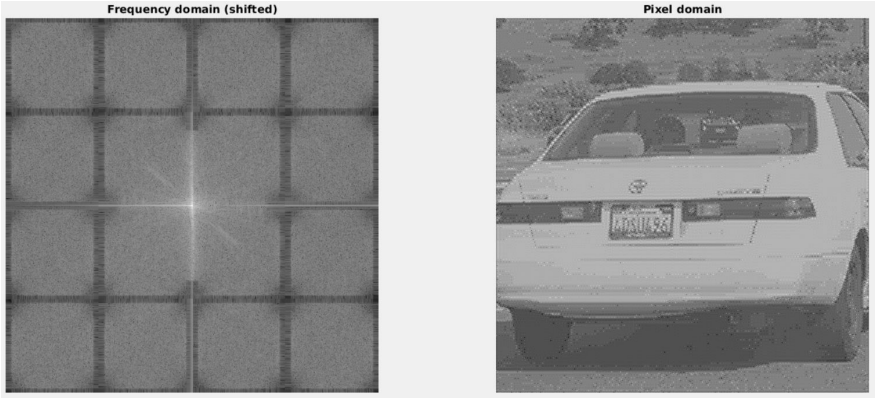


**Fig 12:** Further denoised image (horizontal and vertical lines removed) displayed in the frequency and spectral domain. Compared to the previous denoised image (Fig. 11) an improvement in terms of the elimination of the noisy pattern can be observed for some image regions.

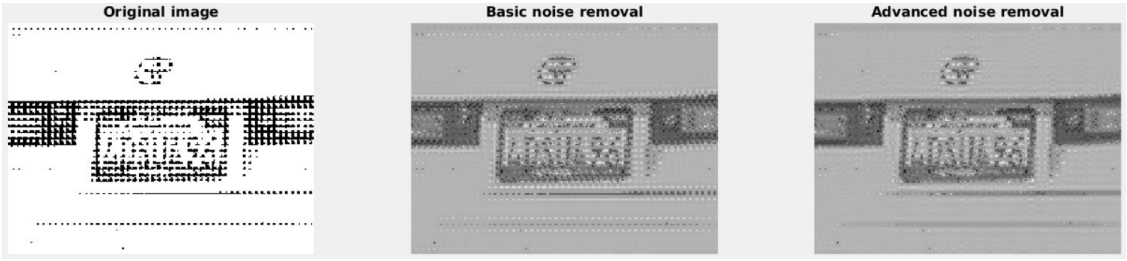


**Fig 13:** Visibility of the car's registration plate across the three images (Fig. 10, 11, 12).

## 3. Lab 3

### *3.1. Exercise 1*

The aim of exercise one of the third lab was to incorporate a watermark in the spectral domain of an image.

- Import the original grayscale *Lena* image (*lena512gray.pgm*). Display the image as well as its first two and last two bit planes (Fig. 14).

- Import the grayscale watermark logo (*warwick512gray.pgm*). Display the original watermark, its binary image and its inverted binary image (Fig. 15).

- Embed an imperceptible watermark in the *Lena* image by replacing its least significant bit (LSB) plane with the inverted binary logo. Display the result and compute its similarity to the original *Lena* image using the Structural Similarity (SSIM) index (Fig. 16).

- Export the watermarked image `WI` and the original `OI` as a JPEG file. Load the images again and compare their LSB planes (Fig. 17).
  **Observation**: The watermark embedded in the `WI` image is no longer visible. Computing the similarity between the two JPEG images' LSB planes results in `SSIM=1` which validates the finding. The conclusion is that JPEG compression irreversibly distorts lower bit planes, hence, erasing the embedded fragile watermark.

- Embed an perceptible watermark in the *Lena* image by replacing its LSB plane with the non-binarised logo. Display the result and compute its similarity to the original *Lena* image using the SSIM index (Fig. 18).


**Fig 14:** The original image in all eight bit planes and in four selected bit planes.

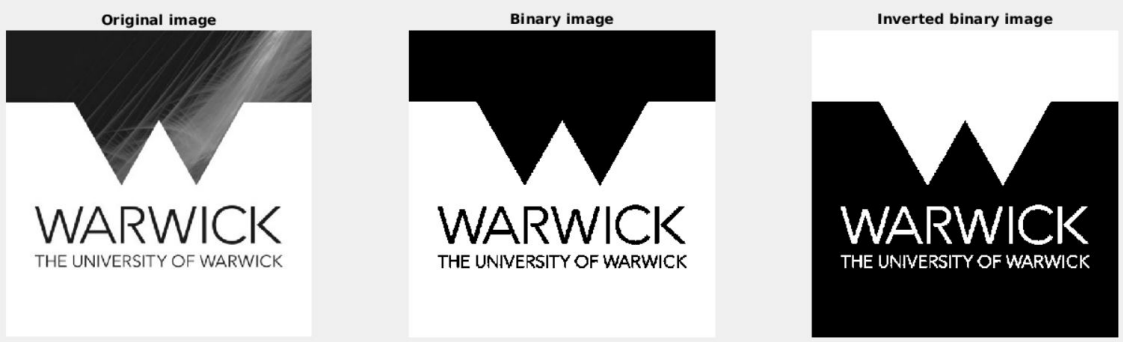


**Fig 15:** The logo image used as a watermark in its original state, binarised and inverted binarised.


**Fig 16:** Image with an imperceptible watermark embedded in its LSB plane. `SSIM=0.9979`.

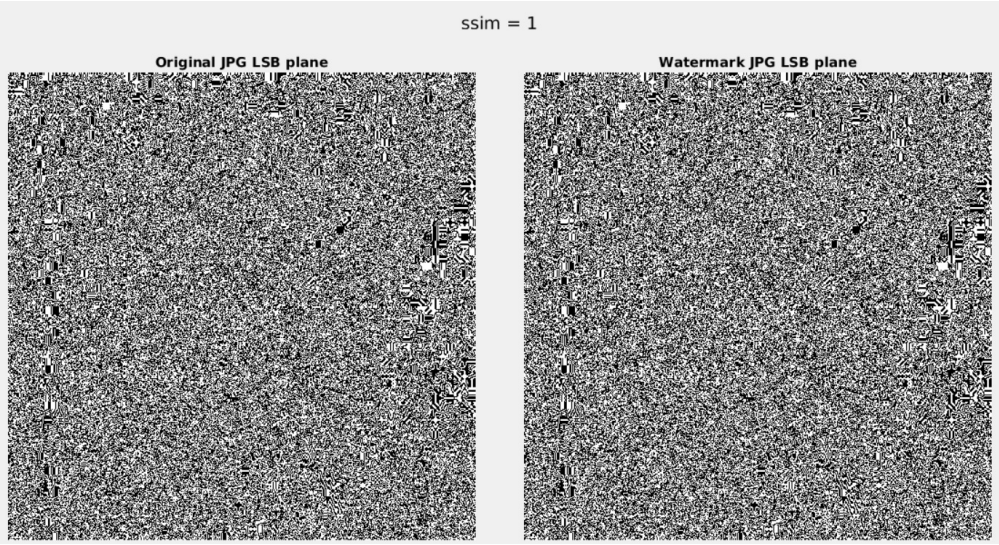


**Fig 17:** LSB planes extracted from JPEG images of the original *Lena* image and the watermarked image.


**Fig 18:** Image with a perceptible watermark embedded in its LSB plane. Watermarks adopted from left to right: non-binarised non-inverted logo (`SSIM=0.6102`), non-binarised inverted logo (`SSIM=0.8390`).

### *3.2. Exercise 2*

The aim of exercise one of the third lab was to incorporate a watermark in the spectral domain of an image.

- Generate a random normalised watermark vector using the `randn` function:
  ```
  w = randn(1,n);
  w = (w-mean(w))/std(w);
  ```
  where $n$ is the length of the watermark vector $\mathbf{w}$.

- Load the *Lena* image and convert it into the frequency domain using the discrete cosine transform (DCT). Find a $3 \times n$ matrix $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n]$ where $\mathbf{h}_i$ is a tuple of the i-th highest coefficient and its position in the DCT matrix.

- Generate a new coefficient $\mathbf{H}'$ matrix by updating the $\mathbf{H}$ matrix as follows:

$$h'_i = h_i(1 + \alpha w_i) \tag{2}$$

  where $h_i$ is the value of $i$-th highest coefficient in $\mathbf{H}$, $w_i$ is the i-th element of $\mathbf{w}$ and $\alpha$ is a constant (in this exercise set to 0.1).

- Create a watermarked image by substituting the altered coefficient values in the DCT matrix according to the positions recorded in $\mathbf{H}'$.

- Convert the image into the spatial domain using the inverse discrete cosine transform (IDCT) and display for selected $n$ values (Fig. 19).

  **Observation**: For the default $\alpha = 0.1$ and examined $n$ range the watermark is imperceptible, however, it becomes the more visible the higher this parameter. Moreover, the difference for distinct $n$ (but equal $\alpha$) is insignificant and the similarity is disproportional to the size of the watermark vector. Both observations are backed by the analysis of calculated SSIM values which do not deviate for a given $\alpha$, however, a change is perceptible for distinct $\alpha$ values.


**Fig 19:** An image with an imperceptible watermark (top row) and a perceptible watermark (bottom row) embedded using DCT coefficients and displayed for a set of parameters.