Advanced Topics in Computer Vision - Lab 5

Leonardo Palacios Fidalgo Ce Xu Zheng

June 5, 2022

Hadamard Transform

In this Lab session we are going to take advantage of the Walsh-Hadamard transformation to add a water mark to the images that we desire. This could be useful to protect the images against the stealing of intellectual property or simply marking the source of them without loosing the capability of recovering the original image if we also have the source of the watermark.

Adding watermark

To add a watermark to the image, the authors of [1] proposed a pipeline were we add the watermark in a gray scale to the luminance of the luminance of the image that we want to watermark in the Hadamard space, and then reconstruct the watermarked image with the new luminance.

To do so, we first need to obtain the gray-scale image of the watermark and transform the source image to the $YCbCr$ color space, were the first Y states for the luminance of the image. Then we need to reescale both the logo and the luminance of the image to the next power of 2 of the greatest dimension for the *fwht* function and finally obtain the Images in the Hadamard space. We can see this preprocessing code in the List.1

Listing 1: Preprocessing of the image

```
%% Adding the watermark

im = imread("photo2.jpg");
wm = imread("water_mark.jpg");
wm = rgb2gray(wm);
wm = double(wm);

PQ = paddedsize( size(im, 1:2), size(wm) , "PWR2");

yuv_im = rgb2ycbcr(im);
y_im = double(yuv_im(:,:,1));
u_im = yuv_im(:,:,2);
v_im = yuv_im(:,:,3);

y_im_augmented = zeros(PQ);
y_im_augmented(1:size(y_im,1), 1:size(y_im,2)) = y_im;

wm_augmented = zeros(PQ);
start_index = int32( (size(im, [1,2])-size(wm))/2 );
h_ind = start_index(1);
w_ind = start_index(2);
wm_augmented( h_ind:h_ind+size(wm, 1)-1, w_ind:w_ind+size(wm, 2)-1 ) = wm;

hadamard_y_im = fwht(y_im_augmented, PQ(1), 'hadamard');
hadamard_w_wm = fwht(wm_augmented, PQ(1), 'hadamard');
```

Once we have the images in the Hadamard space, we just have to mix them by adding a scaled version of the logo to the luminance of the source image. To do so, we need to specify that scaling

factor α that following the suggestions in [1], we are going to use a modified version of the sigmoid function.

$$\alpha = \frac{1}{\pi}\left(arctg(\mu(HI')) + \frac{\pi}{2}\right) \cdot \frac{1}{10^m}, \tag{1}$$

where $\mu(HI')$ is the average value of the source luminance in the Hadamard space, and m is a factor of visibility of the watermark. If $m = 0$, then a watermark prevails the host image, destroying it factually. If $m = 1$, then a watermark visibility is good without destroying the host image (visible watermarking scheme). If $m = 2$, then a watermark become invisible without degrading the quality of a host image (invisible watermarking scheme).

After that, we already have the scaling factor of the watermark and the luminance of the watermarked image in the Hadamard space will be the solution of the element wise addition of the previous luminance and the scaled logo. This could involve a risk of surpassing the maximum limit of the luminance and therefore destroying the image without possibility of a clean reconstruction if the scaling factor α is too high. With the new luminance we only have to concatenate it with the $CbCr$ matrices and transforming it back to an RGB image to obtain the watermarked image.

To test the results of the watermark, we are going to use a photo in the beach as the source image and the logo of Pinterest with a white background changed to grey-scale as the watermark (see Fig.1).

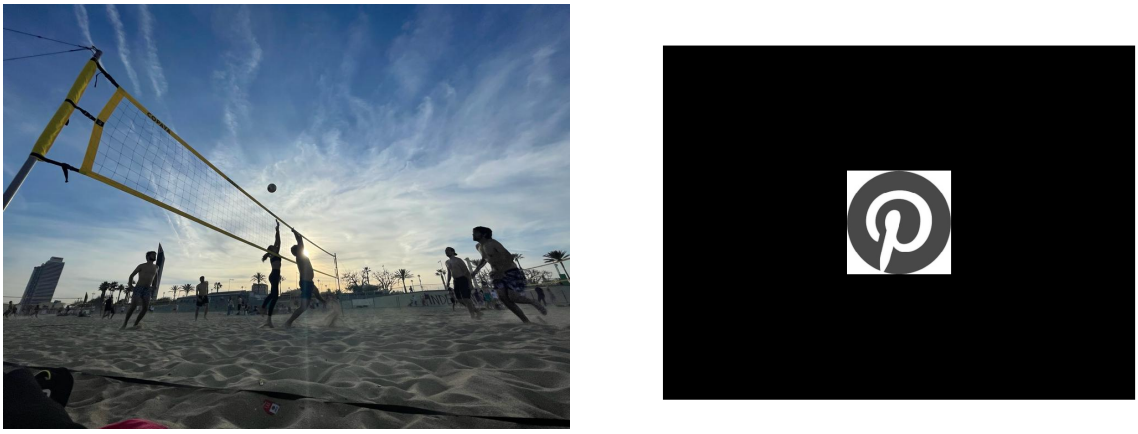


Figure 1: Source image (left) and watermark (right).

In the first place, we are going to see how the visibility factor will affect to the scaling factor by 10 times each, and how this one will affect the appreciation of the watermark.

(a) $m = 0$, $\alpha = 0.5060$. (b) $m = 1$, $\alpha = 0.0506$. (c) $m = 2$, $\alpha = 0.0051$.

Figure 2: Different grades of intensity of the watermark.

As we can see in the Fig.2, the expected behavior of the watermark is accomplished as predicted above. For $m = 0$, the watermark is very strong and saturates the luminance, In the other hand, for $m = 1$, the scaling factor is 10 times lower and we can appreciate the logo in the center of the image. However, if $m = 2$, the watermark is hardly perceptible.

The code for this part will be pretty simple as we can see in the List.2.

Listing 2: Obtaining watermarked image

```
1
2  m = 1;  % controlling parameter if 0 => image destroyed, 1 => good wm, 2 => not ...
       visible wm
3  alpha = 1/pi*( atan(mean(  hadamard_y_im , 'all')) + pi/2 )/(10^m)
4
5  hadamard_y2_im = hadamard_y_im + alpha*hadamard_w_wm;
6
7  y_im2 = ifwht(hadamard_y2_im, PQ(1), 'hadamard');
8  y_im2 = y_im2(1:size(y_im,1), 1:size(y_im,2));
9  yuv_im2 = cat(3,  y_im2 ,u_im,v_im);
10 im2 = ycbcr2rgb( yuv_im2 );
```

Remove the watermark

Once we have a watermarked image, we can extract it only with the source image of the watermark well positioned and the knowledge of which m they used to impact it.

To do so, we have to extract the luminance of the image in the Hadamard space as we have done before, and compute the α as we have done before. Note that if the logo changed the overall luminance considerably, this factor will not be the same as before and therefore, the extraction of it could not be satisfactory.

We can estimate the original luminance in the Hadamard space by extracting the scaled watermark and finally reconstruct the image as we have done before. The code will also be pretty simple in this case thanks to the matlab built in functions as we can see in the List.3.

Listing 3: Remove watermark

```
1  yuv_im2 = rgb2ycbcr(im2);
2  y_im2 = yuv_im2(:,:,1);
3  u_im = yuv_im2(:,:,2);
4  v_im = yuv_im2(:,:,3);
5
6  y_im2_augmented = zeros(PQ);
7  y_im2_augmented(1:size(y_im2,1), 1:size(y_im2,2)) = y_im2;
8
9  hadamard_y_im2 = fwht(y_im2_augmented, PQ(1), 'hadamard');
10
11 alpha = 1/pi*( atan(mean(  hadamard_y_im2 , 'all')) + pi/2 )/(10^m)
12
13 hadamard_y_im3 = hadamard_y_im2 - alpha*hadamard_w_wm;
14
15 y_im3 = ifwht(hadamard_y_im3, PQ(1), 'hadamard');
16 y_im3 = y_im3(1:size(y_im2,1), 1:size(y_im2,2));
17 yuv_im3 = cat(3,y_im3   ,u_im,v_im);
18 im3 = ycbcr2rgb( yuv_im3 );
```

The results of the logo extraction can be seen in the Fig.3.

(a) $m = 0$.

(b) $m = 1$.

(c) $m = 2$.

Figure 3: Different grades of intensity of the watermark.

As we expected, the saturation of the luminance for $m = 0$ is makes the extraction impossible and it will remain a shading where the luminance was saturated. For $m = 1$, the extraction results in a not appreciable watermark, and for the last case, the watermark was not appreciable since the beginning, so we cannot evaluate the result beyond knowing that it didn't worsen the image.

To look for the scaling factor threshold, we tried with different values of the α manually and transmitted to all the extractions. The results are shown in the Fig.4.

(a) $\alpha = 0.1$.

(b) $\alpha = 0.15$.

(c) $\alpha = 0.2$.

Figure 4: Different scaling factors of the watermark.

As we can see here, for $\alpha = 0.1$, the watermark is not appreciable in the recovered image, and for $\alpha = 0.2$, the watermark is clearly visible, but mostly in the more bright part of the original image, where the luminance might have saturated. We have found the threshold more or less in $\alpha = 0.15$, were you can still appreciate the logo if you look closely but might have gone unappreciated if it was not payed too much attention.

Extract the watermark

The last problem that we are going to solve is the extraction of the watermark when you have both the original image and the watermarked one. As we have done before, we can extract the luminance components of both images and transform them to the Hadamard space. Then we can compute the scaling factor with the Eq.1 and the mean of the original image, and finally obtain the logo in the Hadamard space by subtracting the original image luminance to the watermarked one and divide it by the scaling factor. The employed code can be seen in the List.4

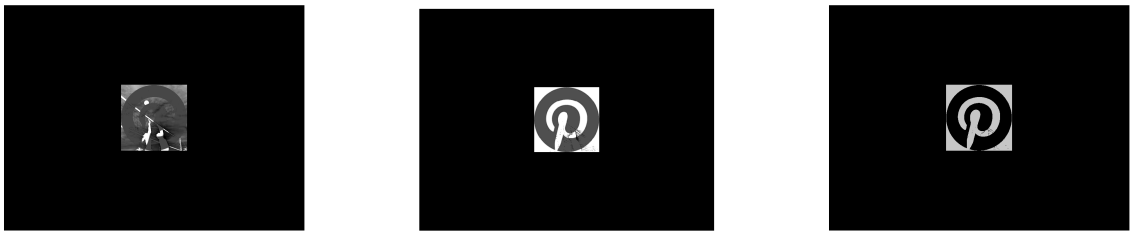
Listing 4: Extracting watermark

```
yuv_im = rgb2ycbcr(im);
y_im = double(yuv_im(:,:,1));
u_im = yuv_im(:,:,2);
v_im = yuv_im(:,:,3);
y_im_augmented = zeros(PQ);
y_im_augmented(1:size(y_im,1), 1:size(y_im,2)) = y_im;
hadamard_y_im = fwht(y_im_augmented, PQ(1), 'hadamard');

yuv_im2 = rgb2ycbcr(im2);
y_im2 = yuv_im2(:,:,1);
y_im2_augmented = zeros(PQ);
y_im2_augmented(1:size(y_im2,1), 1:size(y_im2,2)) = y_im2;
hadamard_y_im2 = fwht(y_im2_augmented, PQ(1), 'hadamard');

% alpha = 1/pi*( atan(mean(  hadamard_y_im , 'all')) + pi/2 )/(10^m)
```

```
16
17 hadamard_extracted_wm = (hadamard_y_im2-hadamard_y_im)/alpha;
18
19 wm_2 = ifwht(hadamard_extracted_wm, PQ(1), 'hadamard');
20 wm_2 = wm_2(1:size(y_im2,1), 1:size(y_im2,2));
```



(a) $m = 0$. (b) $m = 1$. (c) $m = 2$.

Figure 5: Different grades of intensity of the watermark.

As we can see in the Fig.5, the extraction of the quality of the extracted watermark is correlated with the quality of the removal of it. For $m = 0$, the watermark has a great influence of the background, where the lower lights has a greater prevalence in the new logo, and the white background is definitely not recovered. For $m = 1$, the extracted watermark is almost perfect although a small shadow is still casted over the extracted logo, but both the tone of the gray and the white background is distinguishable. Finally, for $m = 2$, the extracted watermark is much darker than the expected, what may mean that the watermarked image had almost no new information of the logo in the luminance after the quantization of it and posterior transformation to RGB.

Here we conclude the Laboratory Session about the Hadamard Transformation and it's application for the watermaking of images.

1 References

[1] Margarita Favorskaya, Eugenia Savchina and Aleksei Popov. Adaptive visible image watermarking based on Hadamard transform. IOP Conf. Series: Materials Science and Engineering 450 (2018) 052003