report.pdf 103062214 李岳容

(1) DCT image compression

1.)how I implement the method

I implement the methods by transfering following formula into program

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N}, \quad \begin{matrix} 0 \le p \le M-1 \\ 0 \le q \le N-1 \end{matrix}$$

$$\alpha_p = \begin{cases} 1/\sqrt{M}, & p = 0 \\ \sqrt{2/M}, & 1 \le p \le M-1 \end{cases} \quad \alpha_q = \begin{cases} 1/\sqrt{N}, & q = 0 \\ \sqrt{2/N}, & 1 \le q \le N-1 \end{cases}$$

$$A_{mn} = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \alpha_p \alpha_q B_{pq} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N}, \quad \begin{matrix} 0 \le m \le M-1 \\ 0 \le n \le N-1 \end{matrix}$$

$$\alpha_p = \begin{cases} 1/\sqrt{M}, & p = 0 \\ \sqrt{2/M}, & 1 \le p \le M-1 \end{cases} \quad \alpha_q = \begin{cases} 1/\sqrt{N}, & q = 0 \\ \sqrt{2/N}, & 1 \le q \le N-1 \end{cases}$$

and use for loop to implement sigma. I use 6 layer forloop to implement the formula

in b.), I transfer RGB into YIQ before implement DCT. And the formula I used to transfer RGB into YIQ is a matrix(refer to p.26 YIQ model)

a.) discuss PSNR value here

-> PSNR = 26.8434 for n = 2, PSNR = 33.2974 for n = 4, PSNR = 304.5931 for n = 8

-> PSNR means how similar the output image is compared with the original image.

-> The larger the PSNR is, the more similar compared with the original image is.

-> The reason why n = 8 has large PSNE is that n = 8 keep more pixel of original picture. However, when n = 2, we will discard some pixel, which result in distortion, and make the PSNR much smaller.

b.) discuss PSNR value here

-> PSNR = 26.8434 for n = 2, PSNR = 33.2974 for n = 4, PSNR = 304.9812 for n = 8

-> PSNR means how similar the output image is compared with the original image.

-> The larger the PSNR is, the more similar compared with the original image is.

-> same as problem a.), n = 8 will keep more pixel, and n = 2 keep less, which leads to the result of PSNR.

c.) compare subproblem a.) with subproblem b.)

-> Both are the same, when it comes to PSNR versus n(for upper left nXn). The Larger the n is, the Bigger the PSN is, which means more similar to the original image.

-> for n = 2, n = 4, the PSNR in a.) and b.) are the same. There's only a little bit

different in PSNR between a.) and b.) for n = 8. Because RGB <-> YIQ just change color but pixel, we won't loss any pixel during transformation.
That's why the PSNR in a.) and b.) is so similar.

DCT image compression(a)

DCT image compression(b)

(2) Image filter

1.)how I implement the method

In gaussian, I use fspecial('gaussian', [3 3], 0.3) to create a mask, and multiple every pixel with mask point to point. In media, every point of output pixel is the median of its surrounding 3X3 matrix.

a.) compares 3X3 Gaussian with 9X9

-> picture using 3X3 Gaussian filter has lots of little and clear white dots. Picture using 9X9 filter also has lots of white dots. However, those dots are not that clear as 3X3. In 9X9, both dots and picture itself are more blur.

b.) compares media filter and Gaussion filter

-> the effect of blur is much better in picture using media filter than gaussian one. In media filter, there's no white dot, and the effect of blur is more smooth and clear than picture using gaussian.


a.) compares 3X3 Gaussian with 9X9


b.) compares media filter and Gaussion filter

(3) Interpolation

1.)how I implement the method

In nearest I use floor(((i-1)/4)+1 floor((j-1)/4)+1 as index to pick up pixel of original image. In bilinear, I used mean of four nearest point which multiple their own weight.

a.) Compute the PSNR with the origial picture.

-> PSNR = 28.1774(same as imresize)

b.) Compute the PSNR with the origial picture.

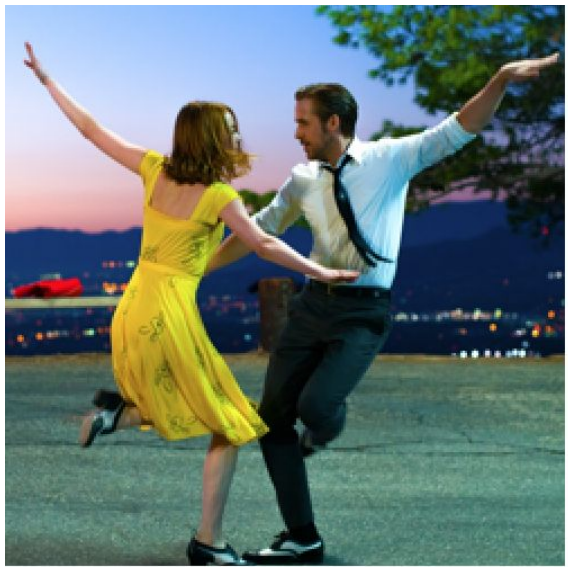
-> PSNR = 24.0196

c.) Compare the result

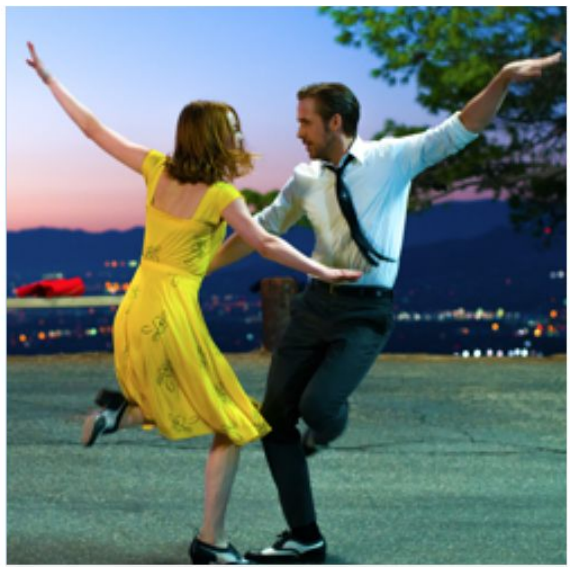
-> PSNR is larger than b.)PSNR, which means picture using nearest_neighbor is more similar to original picture than biliner. However, my method for bilinear is wrong(I guess it is not that good of my index setting), PSNR for bilinear should higher than nearest

-> However, biliner makes the picture more smooth. For neaest_neighbor just

pick one neighbor, it makes 4 pixels same as 1 pixel of original picture, but different with all the other near pixels. However, bilinear makes up its 1 pixel by mixing 4 neighbor, so output will be more smooth.



a.) nearest_neighbor .



b.) bilinear