Digital Image Processing
School of Information Science and Technology
ShanghaiTech University

Assignment 2

An Lihua 2022131001

Due time: 23:59, April 15th, 2023

---

1 Notes

This homework has **100 points** in total.

Please submit your homework to blackboard with a zip file named as **DIP2023_ID_Name_hw2.zip**. The zip file should contain three things: **a folder named 'codes' storing your codes**, **a folder named 'images' storing the original images**, and **your report named as report_ID_Name_hw2.pdf**. The names of your codes should look like **'p1a.m'** (for (a) part of Problem 1), so that we can easily match your answer to the question. Make sure all paths in your codes are relative path and we can get the result directly after running the code. Please answer in English.

Using MATLAB to complete coding assignments is recommended. All core codes are required to be implemented by yourself (without using relevant built-in functions). Make sure your results in the report are the same with the results of your codes. Please explain with notes at least at the key steps of your code.

2 Policy on plagiarism

This is an individual homework. You can discuss the ideas and algorithms, but you can neither read, modify, and submit the codes of other students, nor allow other students to read, modify, and submit your codes. Do not directly copy ready-made or automatically generated codes, or your score will be seriously affected. We will check plagiarism using automated tools and any violations will result in a zero score for this assignment.

3 Problem sets

Problem 1 (37 pts)

(a) Perform DFT to 'Lena.tif' and center the low frequency component (fft and fftshift are not allowed here). Show the frequency spectrum (Hint: log compression helps better display frequency spectrum). (4 pts)

(b) Here, we will use an ideal lowpass filter and a 4th order Butterworth lowpass filter to filter the 'Lena.tif' respectively. Set $D_0 = 30$, show your filter in form of image and the filtered results. (12 pts)

(c) Describe the difference of results in (b), and explain the reason. (4 pts)

(d) Design a highpass filter to extract details of 'Baboon.tif'. (5 pts)
Show the result like this:

Figure 1: Baboon detail

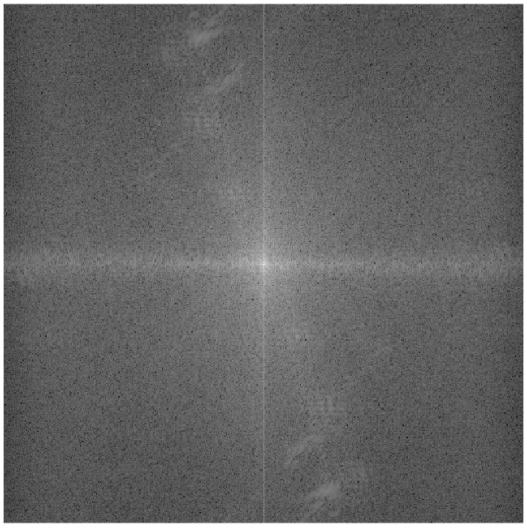


Figure 2: original frequency spectrum

(e) Please design a notch filter to denoise 'Goldhill_noised.tif'. The frequency spectrum of noiseless image is displayed above. Show the frequency spectrum of 'Goldhill_noised.tif' and your denoised result. (12 pts)

Solution:

(a) To realize the 2D DFT algorithm, I use the following equation and create a function **my_fft()**.

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp\left[-2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right]$$

Figure 3: DFT equation.

Similarly, I create a function **my_fftshift()** to center the low frequency component. Having used log compression to show the frequency specturm.

(b) The result of ideal lowpass filter is shown in figure below, as well as the function **my_ILF()**:

For Butterworth filter, the result is shown below:

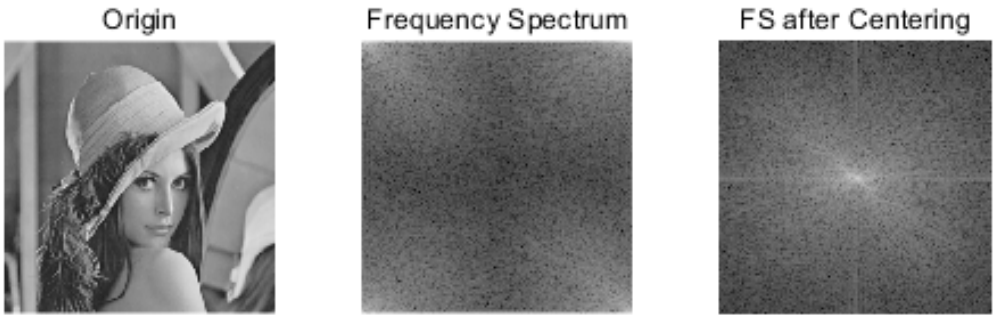


Figure 4: DFT: (a) origin image; (b) frequency spectrum; (c) frequency spectrum after centering.

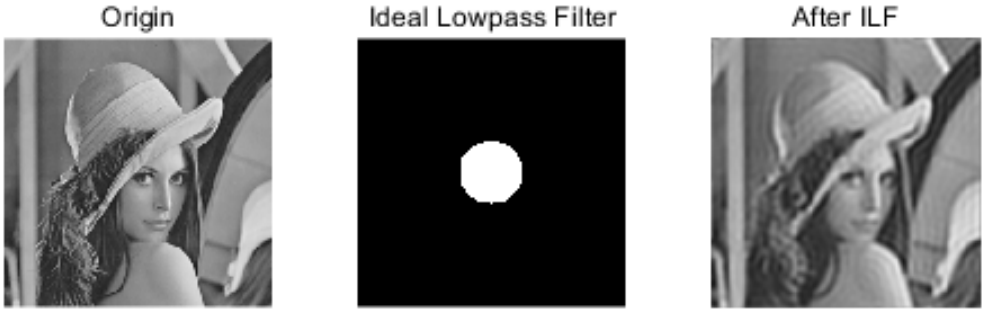


Figure 5: Ideal lowpass filter.

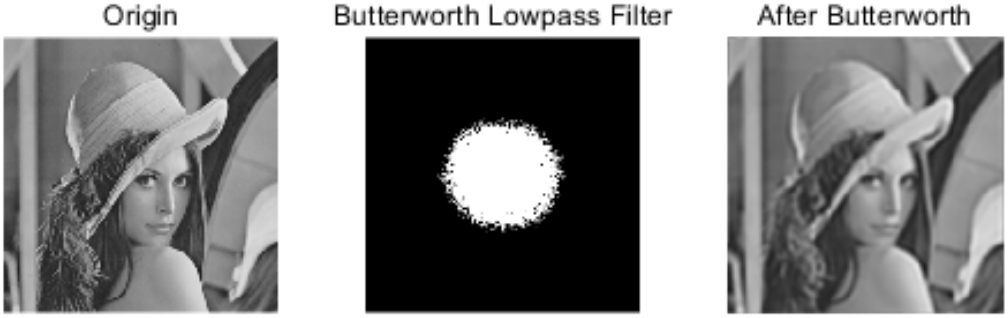


Figure 6: Butterworth filter.

```
function out = my_ILF(image) % ideal lowpass filter
    [M,N] = size(image);
    M1 = ceil(M/2);
    N1 = ceil(N/2);
    out = [M,N];

    D0 = 30;
    for i = 1:M
        for j = 1:N
            distance = sqrt((i-M1)^2+(j-N1)^2);
            if distance <= D0
                h = 1;
            else
                h = 0;
            end
            out(i,j) = h*image(i,j);
        end
    end
end
```

```
function out = my_Butterworth(image, set) % Butterworth
    [M,N] = size(image);
    m = ceil(M/2);
    n = ceil(N/2);
    out = [M,N];

    D0 = 30;
    for i = 1:M
       for j = 1:N
           d = sqrt((i-m)^2+(j-n)^2);
           h = 1/(1+0.414*(d/D0)^(2*set));
           out(i,j)=h*image(i,j);
       end
    end
end
```

Figure 7: (a) Code of ideal lowpass filter; (b) code of Butterworth filter.

(c) An ideal lowpass filter can pass through the filter without attenuation within a radius of D0, and all frequencies outside that radius are completely attenuated. Ideal lowpass filters have the function of smoothing images, but there is a serious ringing phenomenon.

From the function graph of the Butterworth filter, we can see that its transition is not as severe as the ideal lowpass filter. The higher the order, the more severe the filter's transition is, and the more obvious the ringing phenomenon will be.

(d) I use the Butterworth highpass filter to extract the details of the figure, the result is shown below:

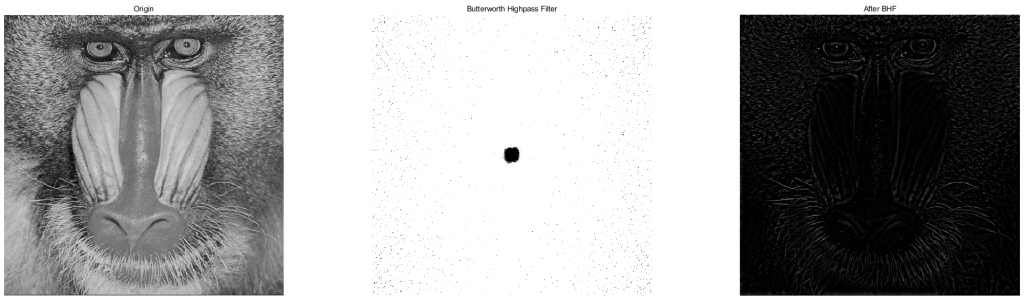


Figure 8: Butterworth highpass filter.

(e) For the notch filter, I have find tree base points to create the filter, like the code of notch filter shown. The points reflected in image is shown in Fig. 10 (c). And from the frequency spectrum in Fig. 11, we can see the filter can perfectly cover the holes in the origin spectrum. And the final result is clear enough as shown in Fig. 10 (d).

```
function out = my_BHF(image, set) % Butterworth Highpass Filter
    [M,N] = size(image);
    m = ceil(M/2);
    n = ceil(N/2);
    out = [M,N];

    D0 = 30;
    for  i=1:M
        for j=1:N
            d=sqrt((i-m)^2+(j-n)^2);
            if (d==0)
                h=0;
            else
                h=1/(1+0.414*(D0/d)^(2*set));
            end
            out(i,j)=h*image(i,j);
        end
    end
end
```

Figure 9: Code of BHF.

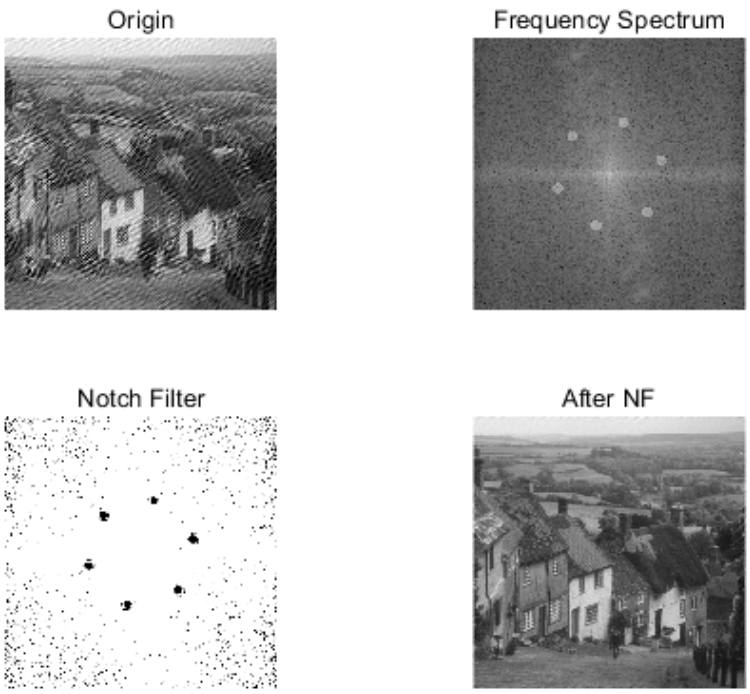


Figure 10: Notch filter: (a) origin image; (b) frequency spectrum of origin image; (c) notch filter; (d) image after notch filter.

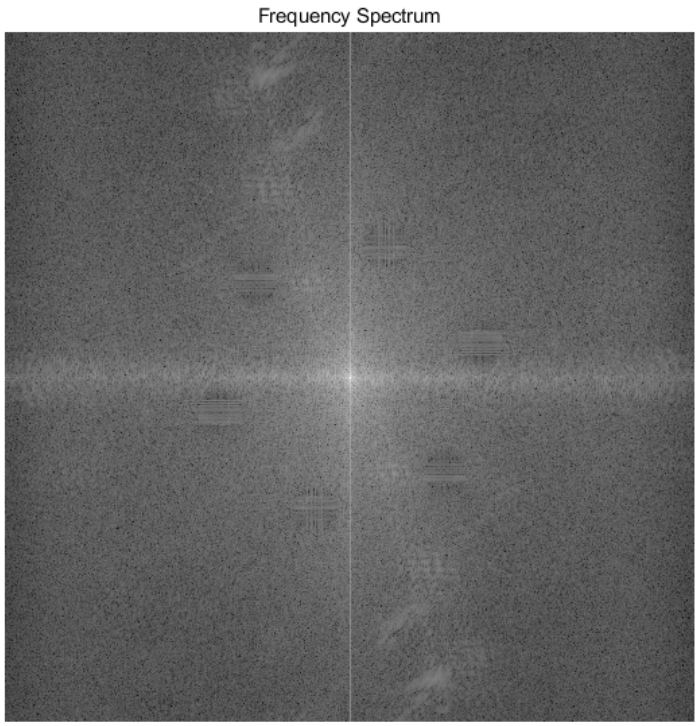


Figure 11: Frequency spectrum of the image after notch filtering.

```
function [NF_Filter,NF_Goldhill] = my_notch(image,set)
    [m,n] = size(image);
    D0=30;
    P=m*2;
    Q=n*2;
    H=ones(P,Q);

    u0 = 140; v0 = 140;
    for x = (-P/2):(P/2)-1
        for y = (-Q/2):(Q/2)-1
          Dk = sqrt((x+u0)^2+(y+v0)^2);
          H(x+(P/2)+1,y+(Q/2)+1) = H(x+(P/2)+1,y+(Q/2)+1) * 1/(1+(D0/Dk)^(2*set));
          Dk = sqrt((x-u0)^2+(y-v0)^2);
          H(x+(P/2)+1,y+(Q/2)+1) = H(x+(P/2)+1,y+(Q/2)+1) * 1/(1+(D0/Dk)^(2*set));
        end
    end

    u0 = -50; v0 = 195;
    for x = (-P/2):(P/2)-1
        for y = (-Q/2):(Q/2)-1
          Dk = sqrt((x+u0)^2+(y+v0)^2);
          H(x+(P/2)+1,y+(Q/2)+1) = H(x+(P/2)+1,y+(Q/2)+1) * 1/(1+(D0/Dk)^(2*set));
          Dk = sqrt((x-u0)^2+(y-v0)^2);
          H(x+(P/2)+1,y+(Q/2)+1) = H(x+(P/2)+1,y+(Q/2)+1) * 1/(1+(D0/Dk)^(2*set));
        end
    end

    u0 = -195; v0 = 50;
    for x = (-P/2):(P/2)-1
        for y = (-Q/2):(Q/2)-1
          Dk = sqrt((x+u0)^2+(y+v0)^2);
          H(x+(P/2)+1,y+(Q/2)+1) = H(x+(P/2)+1,y+(Q/2)+1) * 1/(1+(D0/Dk)^(2*set));
          Dk = sqrt((x-u0)^2+(y-v0)^2);
          H(x+(P/2)+1,y+(Q/2)+1) = H(x+(P/2)+1,y+(Q/2)+1) * 1/(1+(D0/Dk)^(2*set));
        end
    end
    NF_Filter = H .* fftshift(fft2(im2double(im2gray(image)),P,Q));
    NF_Goldhill = real(ifft2(ifftshift(NF_Filter)));
    NF_Goldhill = NF_Goldhill(1:m,1:n);
end
```

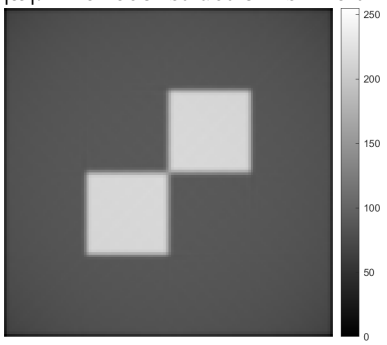
Figure 12: Code of notch filter.

Problem 2 (41 pts)

(a) Given a degraded image 'child_degraded.tif' and the degradation function H_1. Restore the image with full inverse filter and inverse filter with proper cut-off, respectively. Show the results, and explain why full inverse filter works bad. (12 pts)

(b) Given a motion blur degradation function H_2, apply H_2 to 'child.tif' and show the blurred image. Try to restore the blurred image using inverse filter with proper cut-off. Explain why this restoration is more difficult than H_1 (restoration result doesn't need to show). (8 pts)

(c) Given a set of projection data, try to reconstruct the image using filtered back-projection with ramp filter, i.e. $|\omega|$. The reconstruction is like this:



Show your filtered data and the reconstructed image (Hint: function 'imrotate', 'interp', and 'interp2' may be helpful in back-projection, but 'iradon' is not allowed). (15 pts)

(d) Apply hann window to ramp filter, then reconstruct the image again, show the image, describe the function of applying hann window in this question (Hint: hann window can be generated by function 'hann'). (6 pts)

Solution:

(a) Due to the presence of noise N, even if the degradation function H is known, it is not possible to accurately restore the image. Even worse, there may be situations where the degradation function is zero or very small, and the value of N/H is relatively large, making it easy to dominate the estimated value of F. One way to solve this problem is to limit the frequency of filtering. From the spectrum, it can be seen that the value of the high-frequency component is close to 0, while H (0,0) is generally the highest value of H (u, v) in the frequency domain. Therefore, the filtering radius can be shortened to make the passing frequency close to the origin and reduce the probability of encountering zero values.

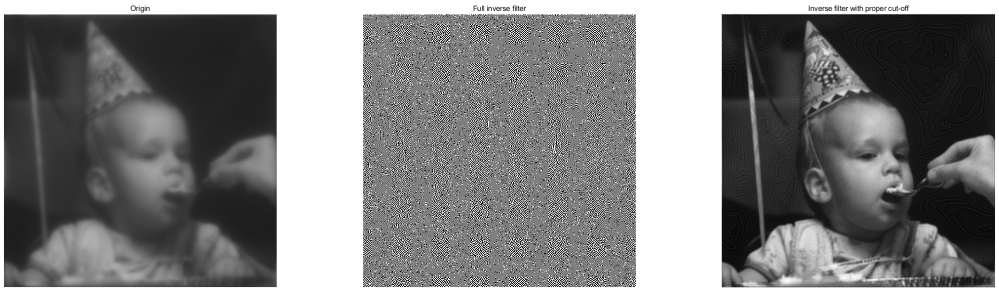


Figure 13: Inverse filter. (a) Origin image, (b) full inverse filter, (c) inverse filter with proper cut-off.

(b) The results are shown below, and (c) is the restored image using the same method as problem 2(a), the result is not good enough.

We can see the frequency spectrum of the two blur degradation functions. For H1 is a circle, it can be easily filterd with proper cut-off, while H2 is linear line cross the whole pixel, it is hard to filter with inverse fitler.

Figure 14: (a) Origin image, (b) motion blurred, (c) inverse filter with proper cut-off.

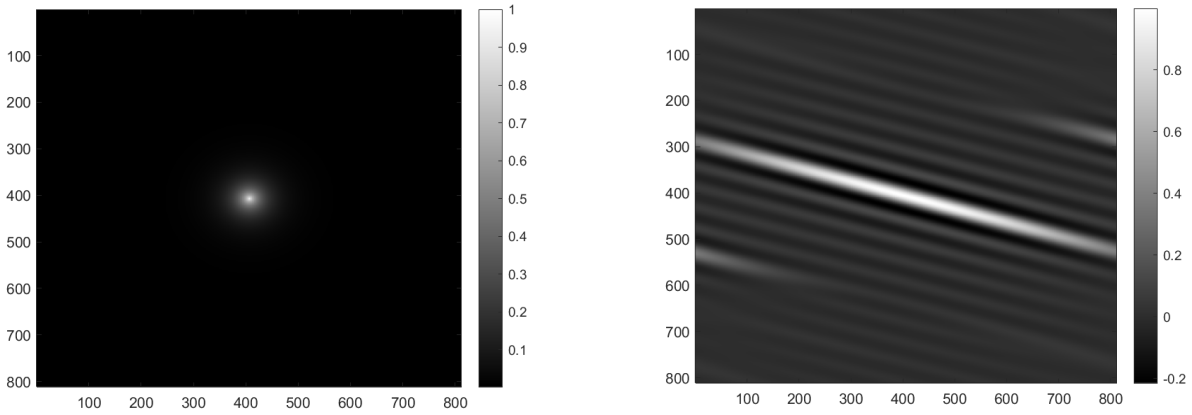


Figure 15: (a) H1; (b) H2.

(c) The process of projection reconstruction is to first perform a one-dimensional Fourier transform on the projection data, and then perform convolution operation with the filter function to obtain the projection data filtered by convolution in each direction. Then, they are backprojected along various directions, evenly distributed to each matrix unit according to their original path, and overlapped to obtain the values of each matrix unit; After appropriate interpolation processing, obtain the cross-sectional image of the scanned object.

The results after FBP are shown below:

(d) Directly using the backprojection algorithm may have two factors that have a negative impact on the experimental results:

1. Inaccurate data reconstruction of images can generate various artifacts.

2. The projected data is naturally discrete, and improper processing can result in significant errors.

So Hann window is used to reduce the artifacts. We can compare the two reconstructed figure in problem (c) and (d), Hann window works well.

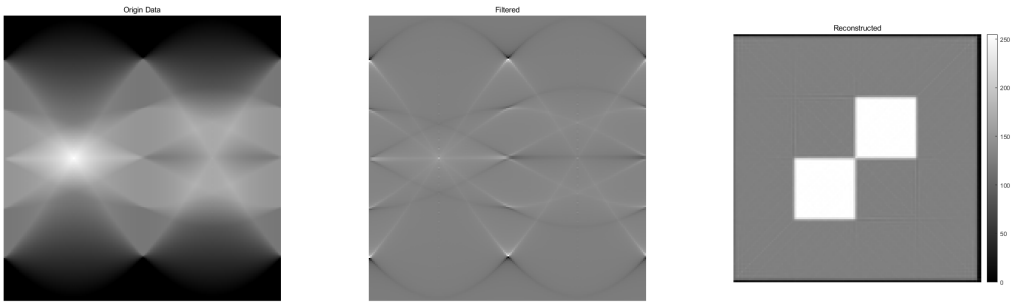


Figure 16: (a) Origin image, (b) filtered image, (c) reconstructed image.

$$w(n) = 0.5\left(1 - \cos\left(2\pi \frac{n}{N}\right)\right), \quad 0 \le n \le N.$$

Figure 17: Hann window.

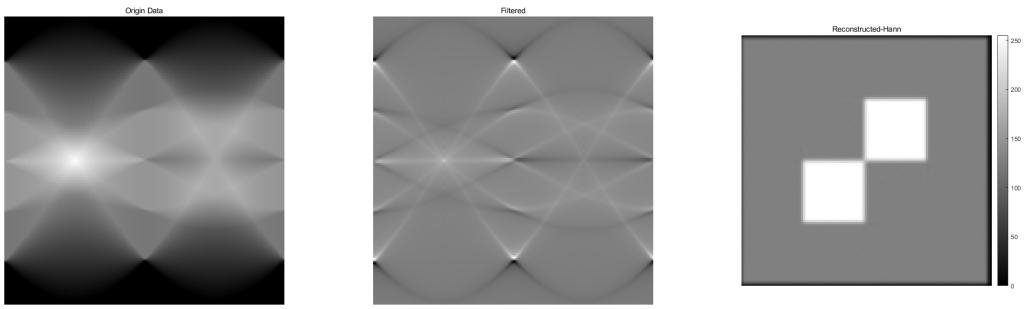


Figure 18: (a) Origin image, (b) filtered image, (c) reconstructed image with Hann window.

Problem 3 (22 pts)

(a) Please convert 'PeppersRGB.tif' from RGB to HSI color space, and then convert it back. Show the image in HSI space and the recovered result in RGB space (Hint: you need to normalize all channels to [0, 1] in HSI space. Also, some exceptional values need to be judged individually). (16 pts)
Results are like this (displayed by imshow(img, [])):

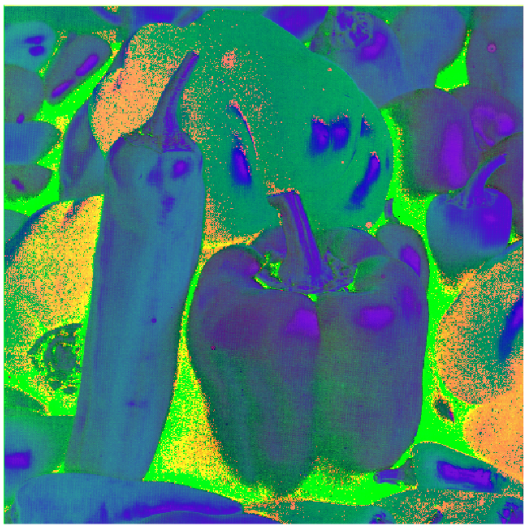


Figure 19: HSI space

Figure 20: RGB space

(b) Choose proper structure element to conduct morphological operation to remove black points inside the object in 'guitar.tif' (6 pts)

Solution:

(a) For the transform equations, from the slice we have:

$$\theta = \arccos\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-G)(G-B)]^{\frac{1}{2}}}\right\}$$

$$H = \begin{cases}\theta, & G \geq B \\ 360-\theta, & G < B\end{cases}$$

$$S = 1 - \frac{3}{R+G+B}[\min(R,G,B)]$$

$$I = \frac{R+G+B}{3} \qquad V = \max(r,g,b);$$

- $0° \leq H < 120°$

  $B = I(1-S), \quad R = I\left[1+\frac{Scos(H)}{\cos(60°-H)}\right], \quad G = 3I-(R+B)$

- $120° \leq H < 240°$

  $R = I(1-S), \quad G = I\left[1+\frac{Scos(H-120°)}{\cos(180°-H)}\right], \quad B = 3I-(R+G)$

- $240° \leq H < 360°$

  $G = I(1-S), \quad B = I\left[1+\frac{Scos(H-240°)}{\cos(300°-H)}\right], \quad R = 3I-(G+B)$

Figure 21: Transform equations: (a) RGB to HSI; (b) HSI to RGB.

The results are shown below:

(b) To remove the black points inside the image, we need to choose **closing** oeration. The structure element is 5×5 is the question. The result is shown below:

Code of closing operation is shown below, closing contains dilation and erosion in order.

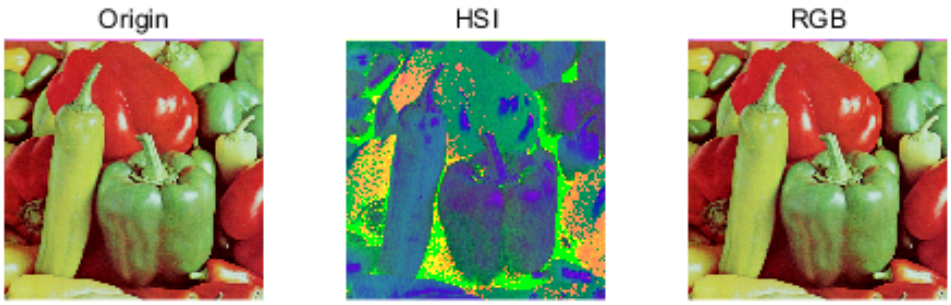


Figure 22: (a) Origin image; (b) HSI image; (c) RGB image converted back.

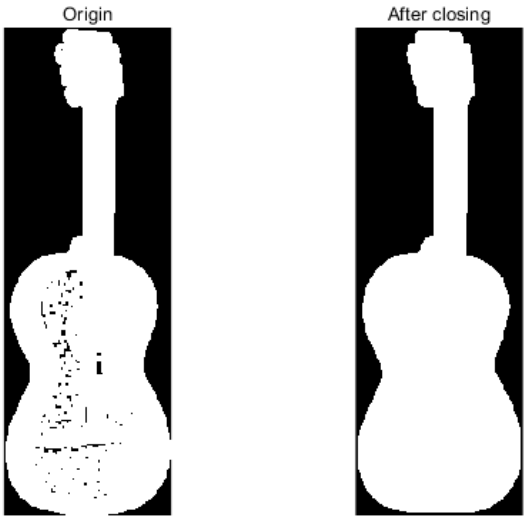


Figure 23: Closing operation.

```
function output = my_closing(image)
    [M,N] = size(image);
    %zero padding
    tmp = zeros(M+4,N+4);
    tmp(3:M+2,3:N+2) = image;

    %Dilation
    Dilation = zeros(M+4,N+4);
    for r = 3:M+2
        for c = 3:N+2
            or0 = bitor(tmp(r-2, c-2), bitor(bitor(tmp(r-2,c-1),tmp(r-2,c)), bitor(tmp(r-2,c+1),tmp(r-2,c+2))));
            or1 = bitor(tmp(r-1, c-2), bitor(bitor(tmp(r-1,c-1),tmp(r-1,c)), bitor(tmp(r-1,c+1),tmp(r-1,c+2))));
            or2 = bitor(tmp(r, c-2), bitor(bitor(tmp(r,c-1),tmp(r,c)), bitor(tmp(r,c+1),tmp(r,c+2))));
            or3 = bitor(tmp(r+1, c-2), bitor(bitor(tmp(r+1,c-1),tmp(r+1,c)), bitor(tmp(r+1,c+1),tmp(r+1,c+2))));
            or4 = bitor(tmp(r+2, c-2), bitor(bitor(tmp(r+2,c-1),tmp(r+2,c)), bitor(tmp(r+2,c+1),tmp(r+2,c+2))));
            Dilation(r, c) = bitor(or0, bitor(bitor(or1, or2), bitor(or3,or4)));
        end
    end

    %Erosion
    Erosion = zeros(M+4,N+4);
    for r = 3:M+2
        for c = 3:N+2
            and0 = bitand(Dilation(r-2, c-2), bitand(bitand(Dilation(r-2, c), Dilation(r-2, c)), bitand(Dilation(r-2, c+1), Dilation(r-2, c+2))));
            and1 = bitand(Dilation(r-1, c-2), bitand(bitand(Dilation(r-1, c), Dilation(r-1, c)), bitand(Dilation(r-1, c+1), Dilation(r-1, c+2))));
            and2 = bitand(Dilation(r, c-2), bitand(bitand(Dilation(r, c), Dilation(r, c)), bitand(Dilation(r, c+1), Dilation(r, c+2))));
            and3 = bitand(Dilation(r+1, c-2), bitand(bitand(Dilation(r+1, c), Dilation(r+1, c)), bitand(Dilation(r+1, c+1), Dilation(r+1, c+2))));
            and4 = bitand(Dilation(r+2, c-2), bitand(bitand(Dilation(r+2, c), Dilation(r+2, c)), bitand(Dilation(r+2, c+1), Dilation(r+2, c+2))));
            Erosion(r, c) = bitand(and0, bitand(bitand(and1, and2), bitand(and3, and4)));
        end
    end

    output = Erosion(3:M+2,3:N+2);
end
```

Figure 24: Code of closing operation.