0.把圖片轉成hsi(code中的img)

```
def feather():   #h的灰階圖跟s的灰階圖做運算 再把它轉成rgb 留下羽毛
    global filename
    img=cv.imread(filename)
    rows = int(img.shape[0])
    cols = int(img.shape[1])
    b, g, r = cv.split(img)
    b = b / 255.0
    g = g / 255.0
    r = r / 255.0
    H, S, I = cv.split(img)
    for i in range(rows):
        for j in range(cols):
            num = 0.5 * ((r[i, j]-g[i, j])+(r[i, j]-b[i, j]))
            den = np.sqrt((r[i, j]-g[i, j])**2+(r[i, j]-b[i, j])*(g[i, j]-b[i, j]))
            theta = float(np.arccos(num/den))
            if den == 0:
                H = 0
            elif b[i, j] <= g[i, j]:
                H = theta
            else:
                H = 2*3.14169265 - theta
            min_RGB = min(min(b[i, j], g[i, j]), r[i, j])
            sum = b[i, j]+g[i, j]+r[i, j]
            if sum == 0:
                S = 0
            else:
                S = 1 - 3*min_RGB/sum
            H = H/(2*3.14159265)
            I = sum/3.0
            img[i, j, 0] = H*255
            img[i, j, 1] = S*255
            img[i, j, 2] = I*255
```

1.分離h s i

```
H_img=img[:,:,0]
S_img=img[:,:,1]
I_img=img[:,:,2]
```

2.分析H和S的灰階圖

H

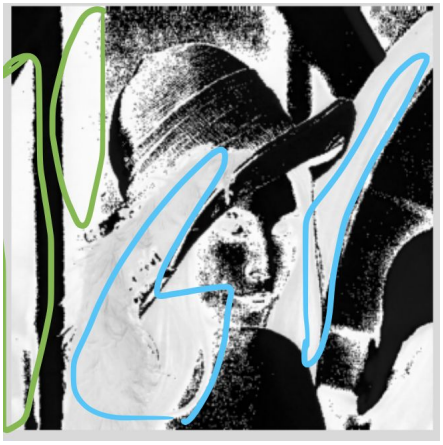
羽毛那塊顏色偏淺

S

 羽毛顏色算中灰色

然後猜測這兩個區域的灰階值

3..不段嘗試哪個範圍比較好



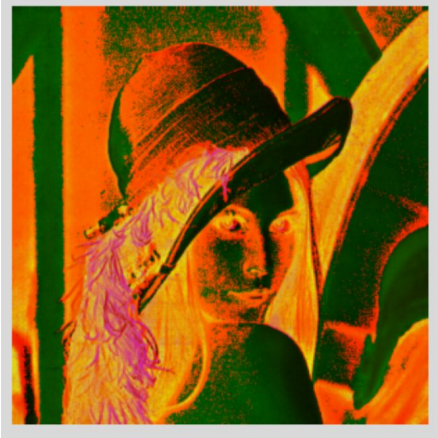
H的圖中，羽毛比較淺，所以我預估灰階值小於200的（比羽毛黑的部分），還有綠色區域灰階值大於225(比羽毛白的部分)，是要去除的部分，讓這些點的i為0，也就是讓他在rgb的情況下會變成黑色。

但是有一大塊區域是與羽毛相似顏色的區域，需要將其去除，所以到S上找範圍

S羽毛的範圍灰階值大概在150~155之間，所以讓點的灰階值在S圖中為150~155以外且在H圖中200~225以外的點去除

```
    h,w = H_img.shape[:2]
    for i in range(h):
        for j in range(w):
            if np.all((H_img[i, j] < 200 or H_img[i, j] > 225 )and ( S_img[i, j] <150 or S_img[i, j] >
155)):
                    I_img[i, j] = 0
```

3.找到一張這樣的hsi圖



```
    img[:,:,0]=H_img
    img[:,:,1]=S_img
    img[:,:,2]=I_img
    rgbimg=HSI2RGB(img)
```

4.寫函數把它轉成rgb

```
def HSI2RGB(hsi_img):     #hsi 轉 rgb
    row = np.shape(hsi_img)[0]
    col = np.shape(hsi_img)[1]
    rgb_img = hsi_img.copy()
    H,S,I = cv.split(hsi_img)
    [H,S,I] = [ i/ 255.0 for i in ([H,S,I])]
    R,G,B = H,S,I
    for i in range(row):
        h = H[i]*2*np.pi
        a1 = h >=0
        a2 = h < 2*np.pi/3
        a = a1 & a2
        tmp = np.cos(np.pi / 3 - h)
        b = I[i] * (1 - S[i])
        r = I[i]*(1+S[i]*np.cos(h)/tmp)
        g = 3*I[i]-r-b
        B[i][a] = b[a]
        R[i][a] = r[a]
        G[i][a] = g[a]
        a1 = h >= 2*np.pi/3
        a2 = h < 4*np.pi/3
        a = a1 & a2
        tmp = np.cos(np.pi - h)
        r = I[i] * (1 - S[i])
        g = I[i]*(1+S[i]*np.cos(h-2*np.pi/3)/tmp)
        b = 3 * I[i] - r - g
        R[i][a] = r[a]
        G[i][a] = g[a]
        B[i][a] = b[a]
        a1 = h >= 4 * np.pi / 3
        a2 = h < 2 * np.pi
        a = a1 & a2
        tmp = np.cos(5 * np.pi / 3 - h)
        g = I[i] * (1-S[i])
        b = I[i]*(1+S[i]*np.cos(h-4*np.pi/3)/tmp)
        r = 3 * I[i] - g - b
        B[i][a] = b[a]
        G[i][a] = g[a]
        R[i][a] = r[a]
    rgb_img[:,:,0] = B*255
    rgb_img[:,:,1] = G*255
    rgb_img[:,:,2] = R*255
    return rgb_img
```