Image Processing
Home work 05
HSI Sharpening

Aqeel Labash
Lecturer: Gholamreza Anbarjafari

19 April 2016

The Images

Fig. 1: The original image and HSV version of the image

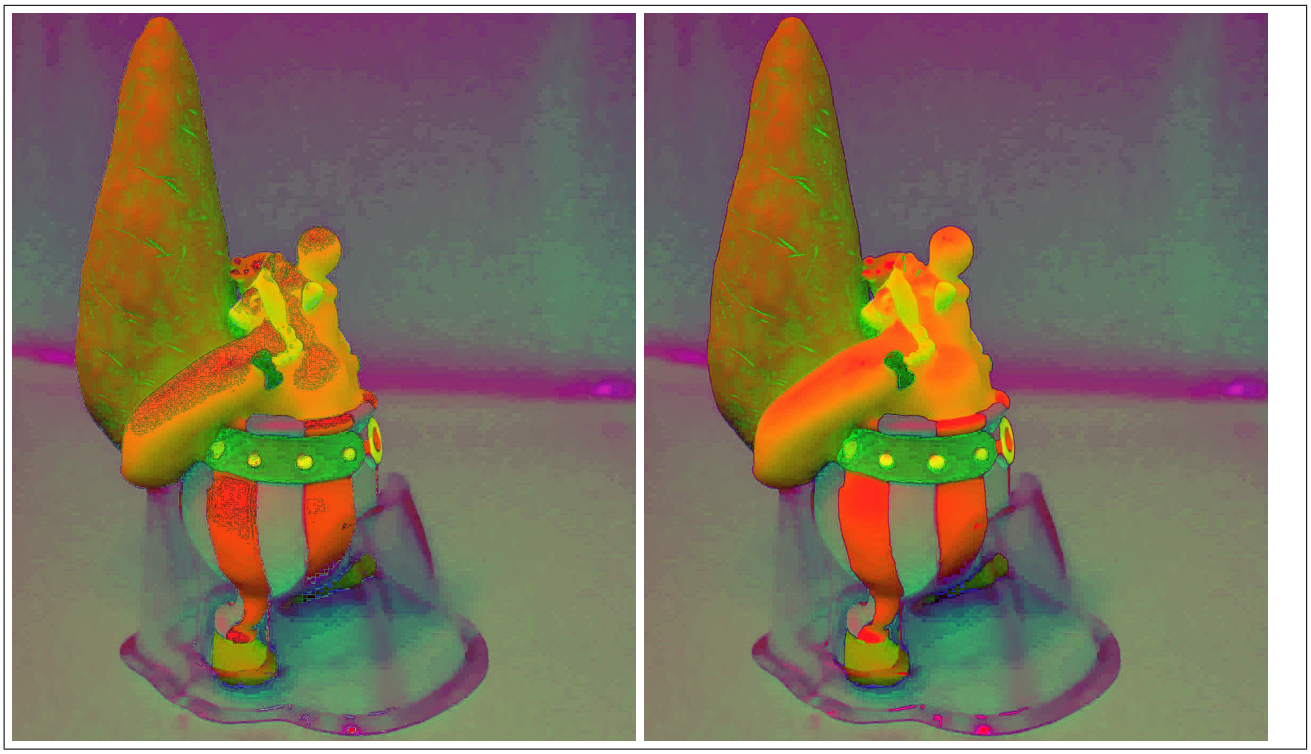

Fig. 2: The left one is layer by layer , right all at once

I used Laplacian sharpening to sharp the images. From the images we can notice that sharpening all of them at once is better I think one by one cause bias at some places where there is varied values in some layers which cause this noisy effect near the arm.

Code

First Code

At first I implemented the HSI method my self by this code :

```
def RGTToHSIV3(img):
    print img.shape
    #Normailization
    img = np.array(img,dtype=np.float)
    thesum =np.sum(img,axis=2,dtype=np.float)
    img[:,:,0] = img[:,:,0]/thesum
    img[:,:,1] = img[:,:,1]/thesum
    img[:,:,2] = img[:,:,2]/thesum
    I = np.sum(img,axis=2)/3
    m = np.min(img,axis=2)
    #By default when divide by zero numpy assigen value of zero.
    S = 1-np.min(img,axis=2)/I
    numinator =img[:,:,2]-0.5*(img[:,:,0]+img[:,:,1]) #(R-1/2*(G+B))
    denum = np.sqrt(np.power(img[:,:,2]-img[:,:,1],2)+(img[:,:,2]-img[:,:,0])*(img[:,:,1]-img
    [:,:,0]))
    theta= np.degrees(np.arccos(numinator/(denum+0.00001)))
    coefg_geqb = img[:,:,1]>=img[:,:,0]
    coefg_g_b = img[:,:,0]>img[:,:,1]
    H = (coefg_geqb*theta+coefg_g_b*(360-theta))/360
    HSI = np.zeros(img.shape,dtype=float)
    HSI[:,:,0] = S
    HSI[:,:,1]=I
    HSI[:,:,2]=H
    return HSI
def HSItoRGB(img):
    R = np.ones((img.shape[0],img.shape[1]))
    G = np.ones((img.shape[0],img.shape[1]))
    B = np.zeros((img.shape[0],img.shape[1]))
    img[:,:,2]*=360
    for i in range (img.shape[0]):
        for j in range(img.shape[1]):
            S = img[i,j,0]
```

```
            I = img[i,j,1]
            H = img[i,j,2]
            if H==0:
                R[i,j]= I+2*I*S
                G[i,j]=I-I*S
                B[i,j] = I-I*S
            elif H>0 and H<120:
                R[i,j]= I + I*S*cos(H)/cos(60-H)
                G = I + I*S*[1 - cos(H)/cos(60-H)]
                B[i,j] = I-I*S
            elif H==120:
                R[i,j]= I-I*S
                G[i,j]=I+2*I*S
                B[i,j] = I-I*S
            elif H>120 and H<240:
                R[i,j] = I - I*S
                G[i,j] = I + I*S*cos(H-120)/cos(180-H)
                B[i,j] = I + I*S*[1 - cos(H-120)/cos(180-H)]
            elif H==240:
                R[i,j] = I - I*S
                G[i,j] = I - I*S
                B[i,j] = I + 2*I*S
            elif H>240 and H<360:
                R[i,j] = I + I*S*[1 - cos(H-240)/cos(300-H)]
                G[i,j] = I - I*S
                B[i,j] = I + I*S*cos(H-240)/cos(300-H)
    print R.shape
```

Second Code

After that I used HSV (I sent an email about it and got a positive response)

```
def readimg(name):
    return cv2.imread(name)

def laplacianSharpening(img):
    laplacian = cv2.Laplacian(img,cv2.CV_64F)
    #showimg(img-laplacian)
    return (img-laplacian)
obelximg = readimg('Obelix.jpg')
obelximg_hsv = cv2.cvtColor(obelximg,cv2.COLOR_BGR2HSV)
cv2.imwrite('obelx_hsv.jpg',obelximg_hsv)
obeliximg_hsv_layerbylayer = obelximg_hsv.copy()
obeliximg_hsv_layerbylayer[:,:,0] = laplacianSharpening(obeliximg_hsv_layerbylayer[:,:,0])
obeliximg_hsv_layerbylayer[:,:,1] = laplacianSharpening(obeliximg_hsv_layerbylayer[:,:,1])
obeliximg_hsv_layerbylayer[:,:,2] = laplacianSharpening(obeliximg_hsv_layerbylayer[:,:,2])

cv2.imwrite('obelx_hsv_sharpen_layerbylayer.jpg',obeliximg_hsv_layerbylayer)
obelximg_hsv = laplacianSharpening(obelximg_hsv)
cv2.imwrite('obelx_hsv_sharpen_all.jpg',obelximg_hsv)
```

Used hints for sharpining from here[1]
Note: All the Code,images,notes , tries , python etc.. files exist on Github

References

[1] Image Gradient