

Lab 2
Pengolahan Citra
Image Enhancement in the Spatial Domain
Jumat, 4 Oktober 2019

A. Transformasi Geometri Pada Citra

Transformasi geometri adalah proses perubahan hubungan spasial antara setiap pixel pada sebuah citra yang memetakan kembali pixel citra input dari posisi awal (x_1, y_1) ke posisi baru (x_2, y_2) pada citra output. Beberapa contoh transformasi geometri pada citra adalah sebagai berikut:

1. Translasi

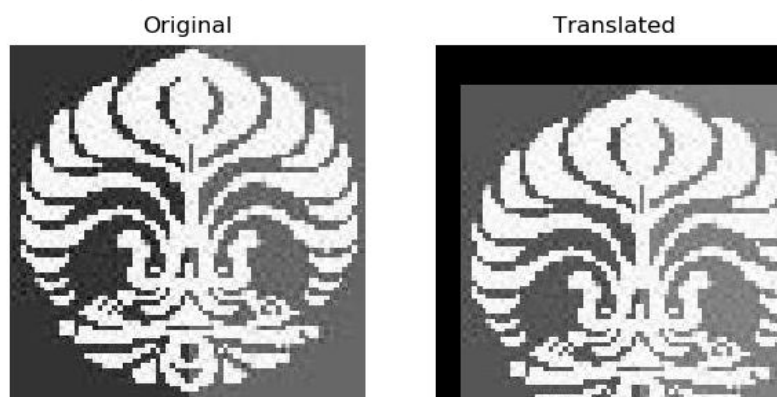
Operasi translasi memindahkan setiap elemen pixel citra input ke posisi baru pada citra output di mana dimensi dari kedua citra tersebut pada umumnya adalah sama.

Contoh fungsi untuk melakukan translasi pada citra grayscale:

```
import numpy as np
def translate(img, x, y):
    height, width = img.shape
    res = np.zeros(img.shape)
    res[y:,x:] = img[:height-y,:width-x]
    return res
```

Untuk menggunakan fungsi tersebut:

```
from skimage import io, color
from matplotlib import pyplot as plt
i1 = color.rgb2gray(io.imread('makara.jpg'))
i2 = translate(i1,15,25)
plt.subplot(1,2,1); plt.imshow(i1, cmap='gray')
plt.title('Original'); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(i2, cmap='gray')
plt.title("Translated"); plt.axis("off")
plt.show()
```



2. Skala

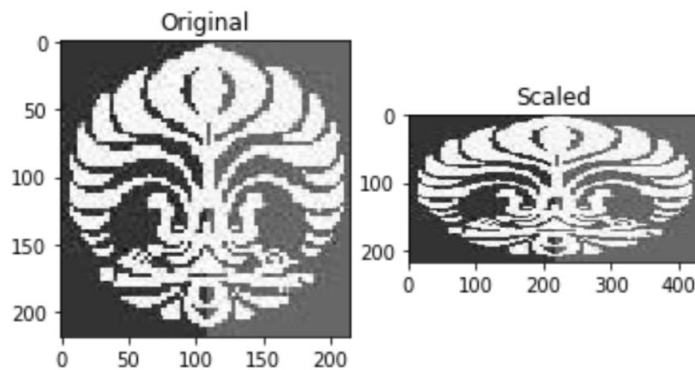
Penskalaan memberikan efek memperbesar atau memperkecil ukuran citra input sesuai dengan variabel penskalaan citranya. Ukuran baru hasil penskalaan didapat melalui perkalian antara ukuran citra input dengan variabel penskalaan.

Contoh fungsi untuk memperbesar ukuran citra grayscale:

```
def perbesar(img, scX, scY):  
    width, height = img.shape  
    r = width * scX  
    c = height * scY  
    newImg = np.zeros((r,c))  
    for i in range(0, r):  
        for j in range(0, c):  
            newImg[i][j] = img[i//scX][j//scY]  
    return newImg
```

Untuk menggunakan fungsi tersebut:

```
i2 = perbesar(i1, 1, 2)  
plt.subplot(1,2,1); plt.imshow(i1, cmap='gray')  
plt.title('Original')  
plt.subplot(1,2,2); plt.imshow(i2, cmap='gray')  
plt.title("Scaled")  
plt.show()
```



3. Rotasi

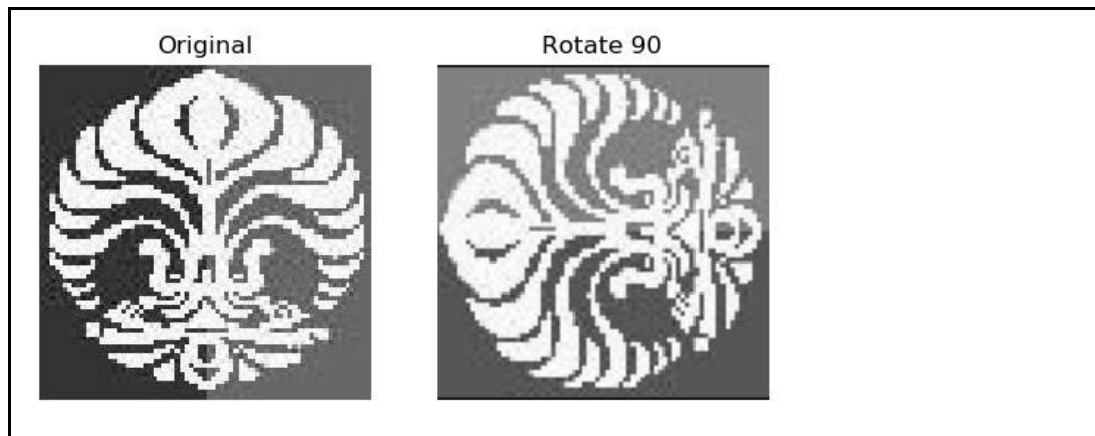
Rotasi merupakan suatu transformasi geometri yang memindahkan nilai-nilai pixel dari posisi awal menuju ke posisi akhir yang ditentukan melalui nilai variabel rotasi sebesar θ° terhadap sudut 0° atau garis horizontal dari citra.

Sintaks:

```
Y = rotate(X,  $\theta$ )
```

Contoh:

```
from skimage.transform import rotate  
  
i1_rotate = rotate(i1, 90)  
plt.subplot(1,2,1); plt.imshow(i1, cmap='gray')  
plt.title('Original'); plt.axis("off")  
plt.subplot(1,2,2); plt.imshow(i1_rotate, cmap='gray')  
plt.title("Rotate 90"); plt.axis("off")  
plt.show()
```



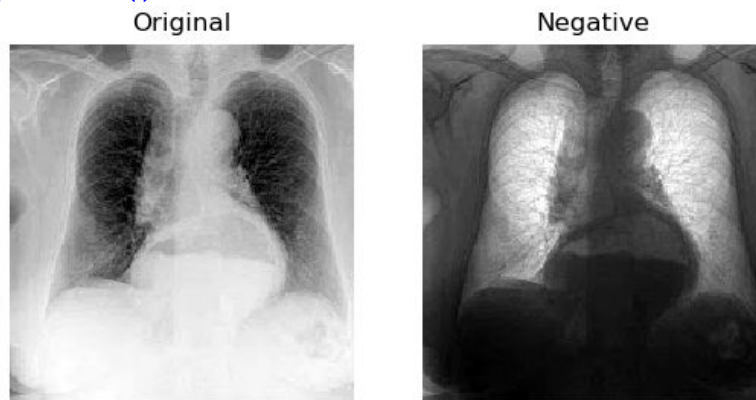
B. Image Enhancement (Spatial Domain)

1. Point Processing

Pemrosesan citra yang operasinya hanya melibatkan satu piksel saja, tidak memperhatikan ketetanggaan.

a. Image Negative

```
i1 = io.imread('rontgen.jpg')
i2 = 255-i1
plt.subplot(1,2,1); plt.imshow(i1)
plt.title('Original'); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(i2)
plt.title('Negative'); plt.axis("off")
plt.show()
```

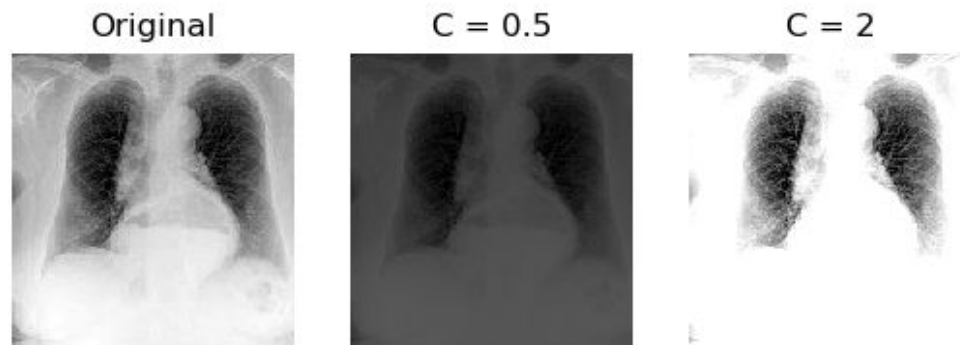


b. Log Transformation

Menggunakan fungsi transformasi $s = c \cdot \log(1 + r)$, dimana c adalah konstanta, dan r adalah citra yang akan ditransformasi. Transformasi Log memetakan suatu citra dengan range warna sempit menjadi lebih lebar pada citra outputnya. Tujuannya untuk ekspansi nilai pixel gelap dan kompresi nilai pixel terang.

```
a = i1/255
c1 = 0.5
c2 = 2
f1 = c1*np.log(1 + (a))
f2 = c2*np.log(1 + (a))
```

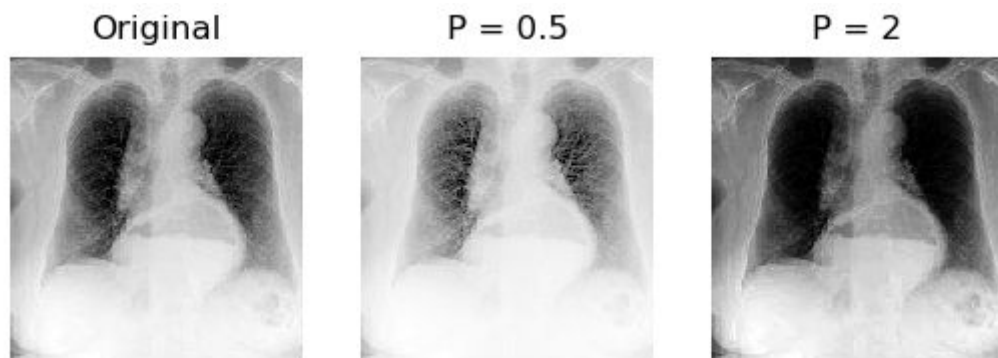
```
plt.subplot(1,3,1); plt.imshow(i1)
plt.title('Original'); plt.axis("off")
plt.subplot(1,3,2); plt.imshow(f1)
plt.title('C = 0.5'); plt.axis("off")
plt.subplot(1,3,3); plt.imshow(f2)
plt.title('C = 2'); plt.axis("off")
plt.show()
```



c. Gamma (Power) Transformation

Gamma transformation hampir sama dengan log transformation. Bentuk umumnya $s = cr^p$ untuk input r dan output s , c dan p adalah konstan.

```
a = i1/255
c = 1
p1 = 0.5
p2 = 2
f1 = c*(a**p1)
f2 = c*(a**p2)
plt.subplot(1,3,1); plt.imshow(i1)
plt.title('Original'); plt.axis("off")
plt.subplot(1,3,2); plt.imshow(f1)
plt.title('P = 0.5'); plt.axis("off")
plt.subplot(1,3,3); plt.imshow(f2)
plt.title('P = 2'); plt.axis("off")
plt.show()
```



d. Contrast Stretching

Contrast stretching mengembangkan range level intensitas pixel yang tadinya terbatas sehingga memiliki range intensitas penuh. Contoh fungsi yang dapat digunakan adalah citra masukan yang gray level-nya tidak penuh dari 0 –255 (low contrast) diubah menjadi citra yang gray level nya berkisar penuh dari 0 –255 (high contrast).

```
i3 = io.imread('rontgenggelap.JPG')
mn = min(i3.flatten())
mx = max(i3.flatten())
b = int(np.floor(255 / (mx - mn)))
i3_cs = (i3 - mn) * b

plt.subplot(1,2,1); plt.imshow(i3)
plt.title('Original'); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(i3_cs)
plt.title('Contrast Stretching'); plt.axis("off")
plt.show()
```

Original



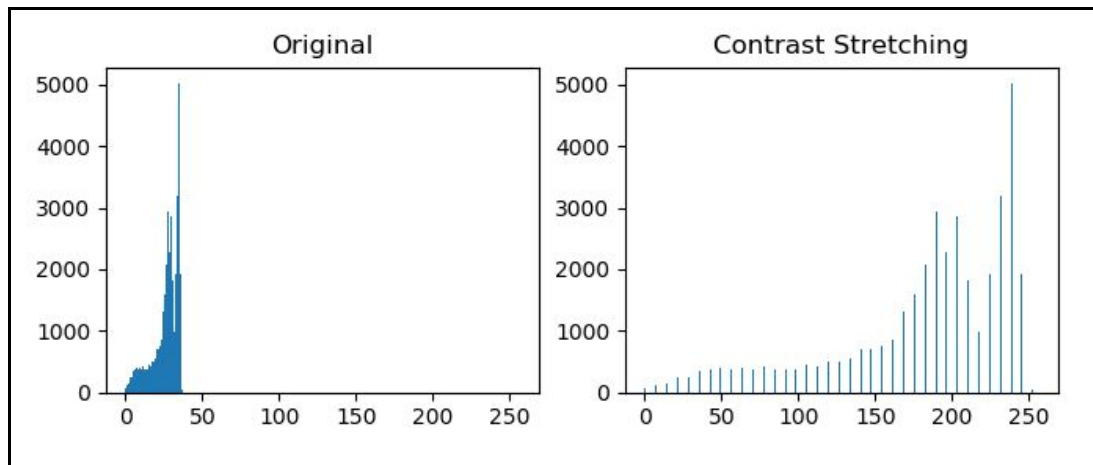
Contrast Stretching



e. Image Histograms

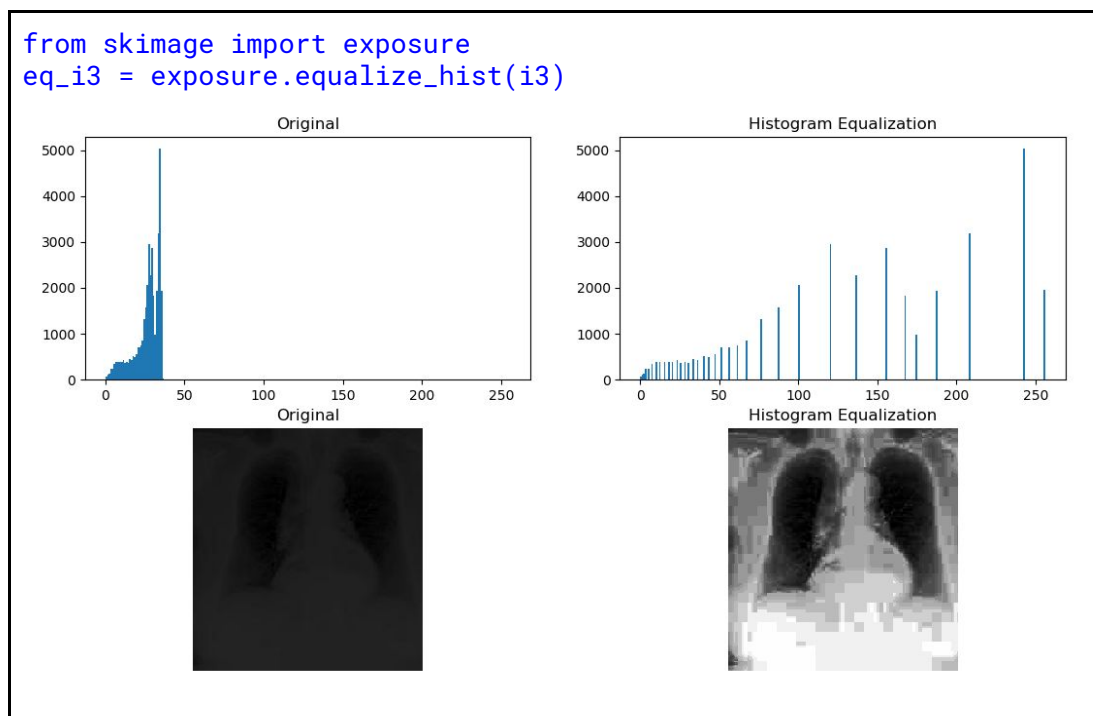
Histogram merupakan suatu teknik domain spasial yang mampu memberikan deskripsi global pada tampilan citra. `plt.hist(var)` digunakan untuk menghitung histogram pada citra grayscale.

```
from skimage import util
gray = util.img_as_ubyte(color.rgb2gray(i3))
plt.subplot(1,2,1); plt.hist(gray.flatten(), 256, range=(0,256))
gray2 = util.img_as_ubyte(color.rgb2gray(i3_cs))
plt.subplot(1,2,2); plt.hist(gray2.flatten(), 256, range=(0,256))
plt.show()
```



f. Histogram Equalization

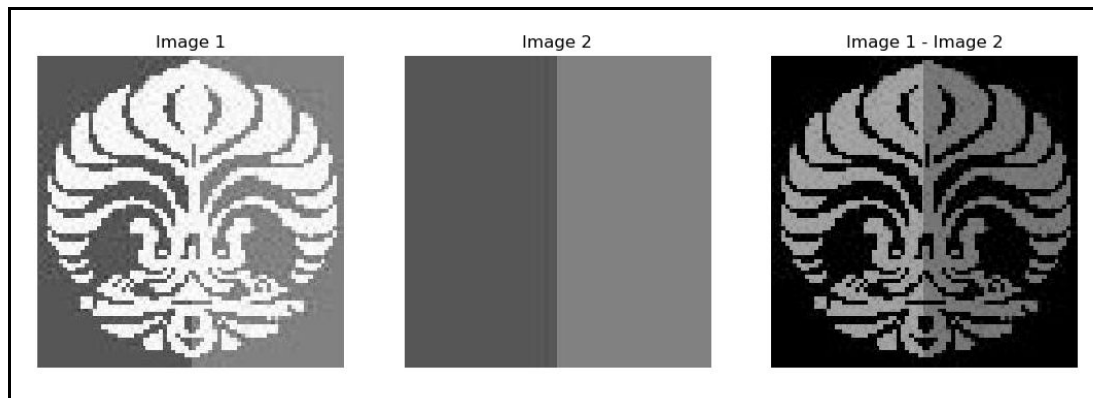
Mengubah pemetaan gray level agar sebarannya (kontrasnya) lebih menyebar pada kisaran 0-255. Histogram processing digunakan untuk mengubah bentuk histogram agar pemetaan gray level pada citra juga berubah. Pada scikit-image, kita bisa menggunakan method `equalize_hist(img)` pada library **exposure**.



g. Image Subtraction

Dilakukan jika kita ingin mengambil bagian tertentu saja dari citra.

```
i1 = io.imread('makara.jpg')
i2 = io.imread('background.jpg')
i_subs = np.subtract(i1, i2, dtype='int16')
```



2. Mask Processing

Mask Processing sering juga disebut dengan Image Filtering yaitu melakukan operasi terhadap suatu jendela ketetanggaan pada citra. Filter pada dasarnya adalah sebuah metode untuk meredam atau menghilangkan noise pada citra digital. Jenis filter bermacam-macam dan fungsi serta efeknya juga berbeda-beda pula.

a. Smoothing Filter

Smoothing bertujuan untuk menekan gangguan (noise) pada citra. Gangguan pada citra umumnya berupa variasi intensitas suatu piksel yang tidak berkorelasi dengan piksel-piksel tetangganya.

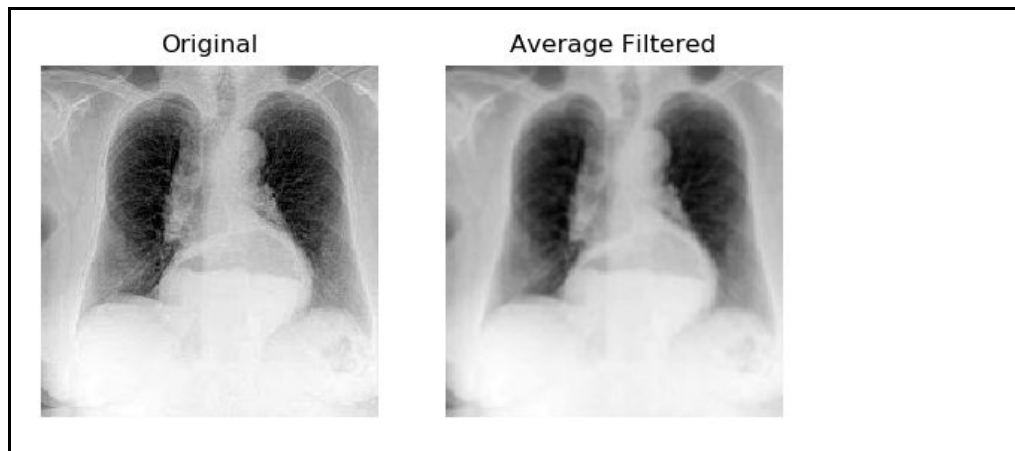
1) Linear Filter

Filter spasial linear adalah filter yang bekerja dengan cara korelasi atau konvolusi yang terdiri dari perkalian setiap pixel di lingkungan dengan koefisien yang sesuai dan menjumlahkan hasil untuk mendapatkan respon pada setiap titik (x, y) . Mekanisme dari spasial filtering hanya terdiri dari memindahkan filter mask dari titik satu ke titik lainnya pada suatu citra. Berikut beberapa macam filter linear yang dibahas pada Lab ini:

- Average Filter

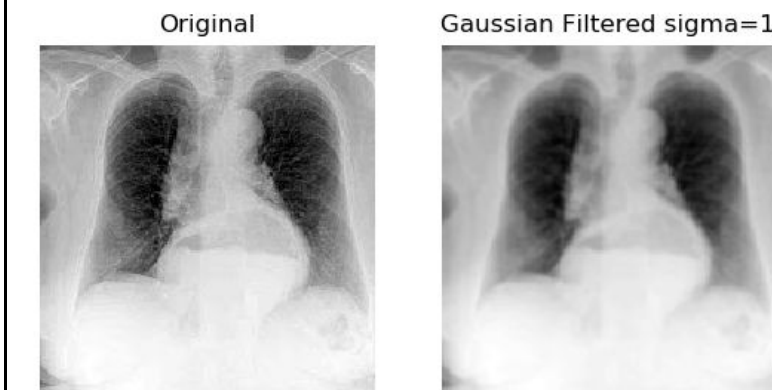
Average filter bekerja dengan cara mengganti nilai suatu piksel pada citra asal dengan nilai rata-rata dari piksel tersebut dan lingkungan tetangganya.

```
from skimage import filters, morphology
gray = color.rgb2gray(io.imread('rontgen.jpg'))
fi = filters.rank.mean(gray, selem=morphology.square(3))
plt.subplot(1,2,1);plt.imshow(i1,cmap='gray',vmin=0,vmax=255)
plt.subplot(1,2,2);plt.imshow(fi,cmap='gray',vmin=0,vmax=255)
plt.show()
```



- Gaussian Filter

```
fi = util.img_as_ubyte(filters.gaussian(gray, sigma=1))
```



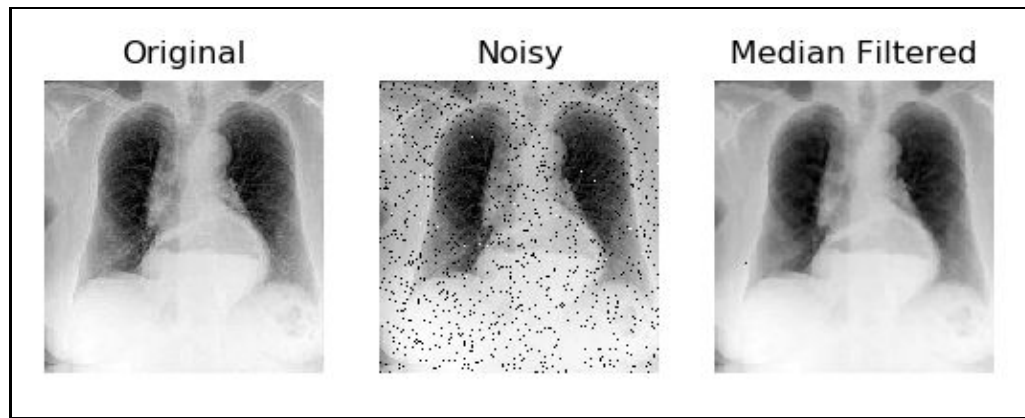
2) Non Linear Filter

Filter spasial non-linier atau biasanya disebut juga dengan filter statistik berdasar urutan adalah filter yang respon nya didasarkan pada urutan atau rangking piksel yang ada dalam citra yang dicakup oleh area filter dengan menggantikan nilai dari piksel yang berada di tengah digantikan dengan nilai hasil pengurutan atau perangkingan tersebut. Filter non-linear memiliki lebih banyak keunggulan dibandingkan dengan filter linear pada ukuran jendela filter yang sama. Berikut beberapa contoh filter non-linear:

- Median Filter

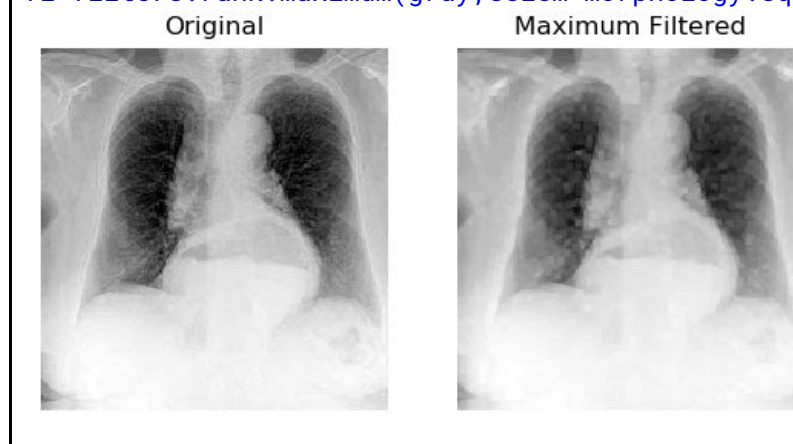
Median filter bekerja dengan mengganti nilai suatu piksel pada citra asal dengan nilai median dari piksel tersebut dan lingkungan tetangganya. Dibandingkan dengan neighborhood averaging, filter ini lebih tidak sensitif terhadap perbedaan intensitas yang ekstrim.

```
noise=util.img_as_ubyte(util.random_noise(gray,
mode='s&p', salt_vs_pepper=0.02))
fi=filters.rank.median(noise, selem=morphology.square(3))
```

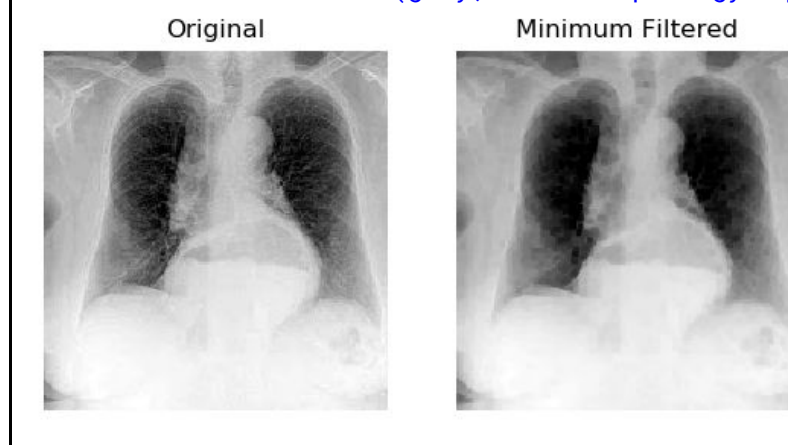
- **Maximum Filter**

```
fi=filters.rank.maximum(gray,selem=morphology.square(3))
```



- **Minimum Filter**

```
fi=filters.rank.minimum(gray,selem=morphology.square(3))
```

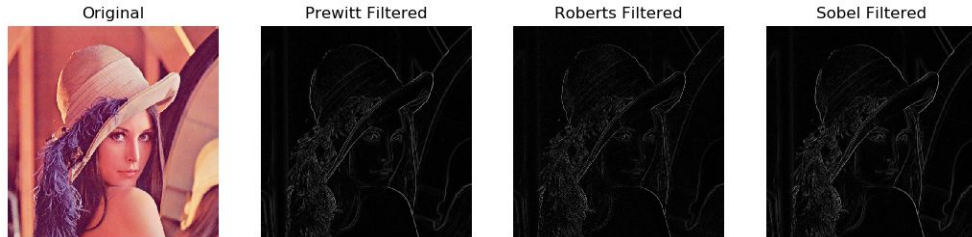


b. Sharpening Filter

Sharpening citra adalah memperjelas tepi pada objek di dalam citra. Sharpening citra merupakan kebalikan dari smoothing citra karena operasi ini menghilangkan bagian citra yang lembut.

- **Roberts, Prewitt, Sobel (edge detection)**

```
gray = color.rgb2gray(i1)
ed1 = util.img_as_ubyte(filters.prewitt(gray))
ed2 = util.img_as_ubyte(filters.roberts(gray))
ed3 = util.img_as_ubyte(filters.sobel(gray))
```



```
# Sharpening Image
gray = color.rgb2gray(io.imread('rontgen.jpg'))
fi = filters.rank.mean(gray, selem=morphology.square(3))
sh = filters.unsharp_mask(fi, radius=3)
```



c. Color Image Processing

- **Color transformation**

Color transformation yang di maksud di sini adalah transformasi dalam 1 buah color model saja.

- **RGB**

```
RGB = io.imread('jakarta.jpg')
R = RGB[:, :, 0]
G = RGB[:, :, 1]
B = RGB[:, :, 2]
plt.subplot(2,2,1);plt.imshow(RGB)
plt.subplot(2,2,2);plt.imshow(R,cmap='gray',vmin=0,vmax=255)
plt.subplot(2,2,3);plt.imshow(G,cmap='gray',vmin=0,vmax=255)
plt.subplot(2,2,4);plt.imshow(B,cmap='gray',vmin=0,vmax=255)
plt.show()
```

RGB Image



Red Component



Green Component



Blue Component



- CMY

```
C = 1 - util.img_as_float(R)
M = 1 - util.img_as_float(G)
Y = 1 - util.img_as_float(B)
CMY = np.zeros(RGB.shape)
CMY[:, :, 0] = C; CMY[:, :, 1] = M; CMY[:, :, 2] = Y
plt.subplot(2,2,1); plt.imshow(CMY)
plt.subplot(2,2,2); plt.imshow(C, cmap='gray')
plt.subplot(2,2,3); plt.imshow(M, cmap='gray')
plt.subplot(2,2,4); plt.imshow(Y, cmap='gray')
plt.show()
```

CMY Image



C Component



M Component

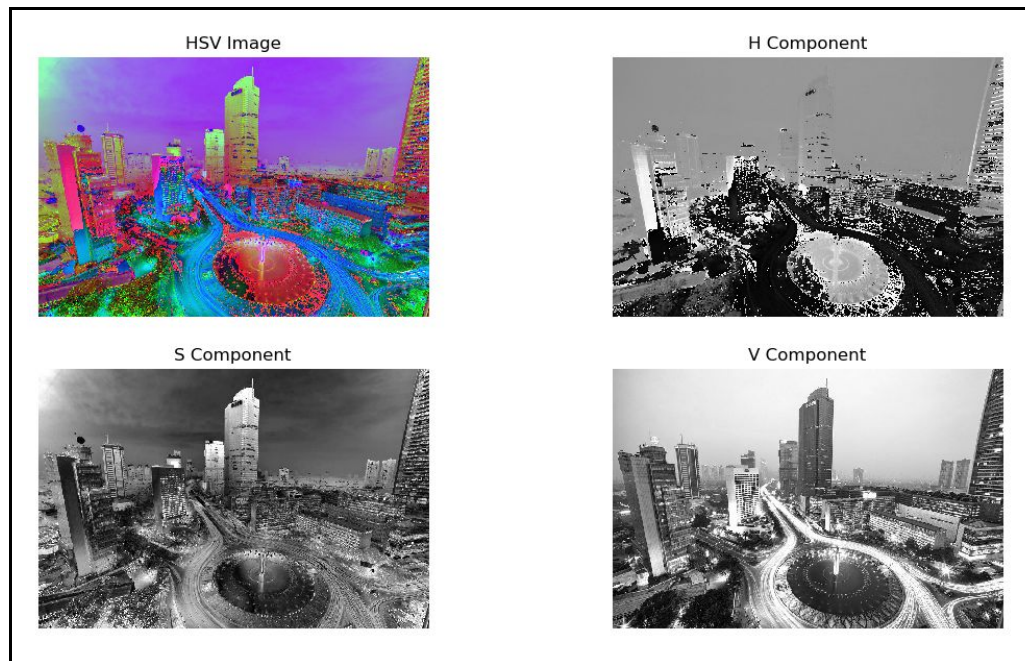


Y Component



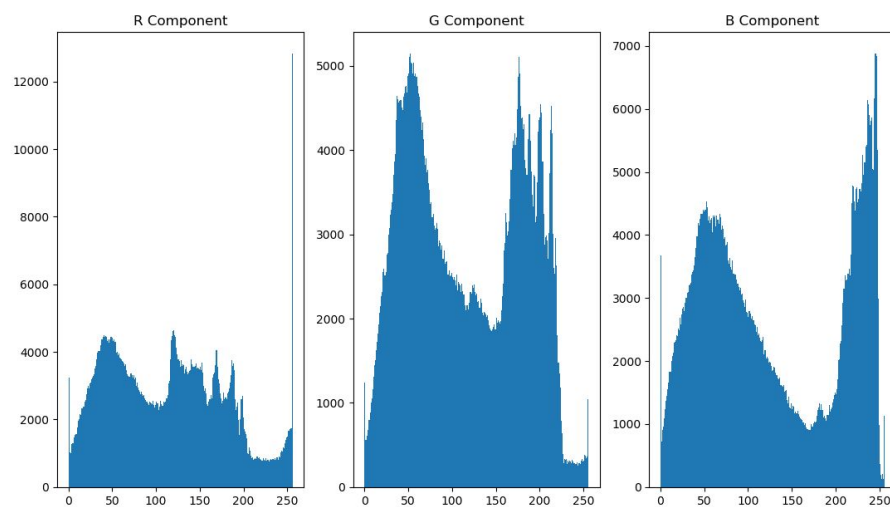
- HSV

```
HSV = color.rgb2hsv(RGB)
H = HSV[:, :, 0]
S = HSV[:, :, 1]
V = HSV[:, :, 2]
```

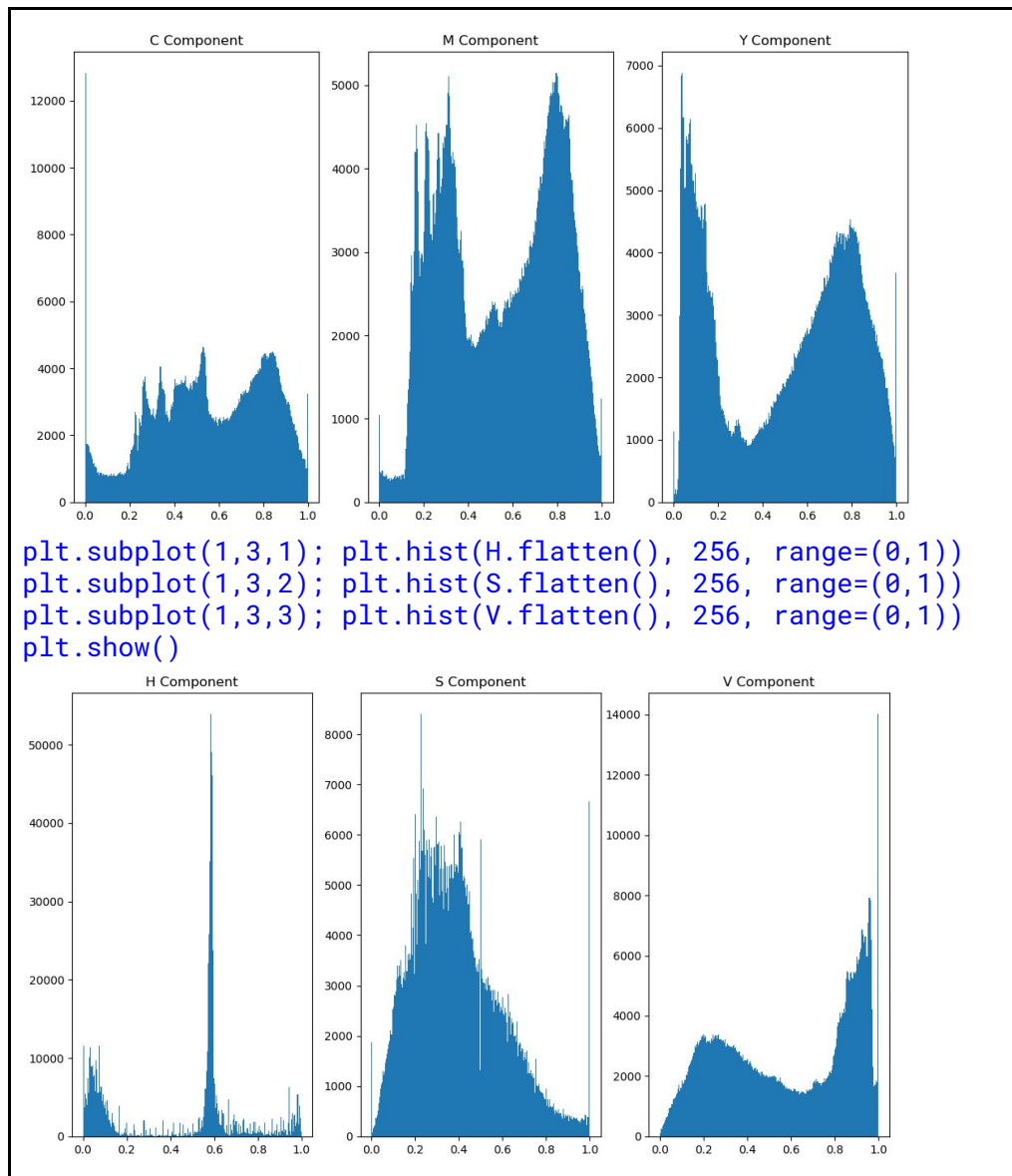


- Histogram Processing

```
plt.subplot(1,3,1); plt.hist(R.flatten(), 256, range=(0,256))
plt.subplot(1,3,2); plt.hist(G.flatten(), 256, range=(0,256))
plt.subplot(1,3,3); plt.hist(B.flatten(), 256, range=(0,256))
plt.show()
```



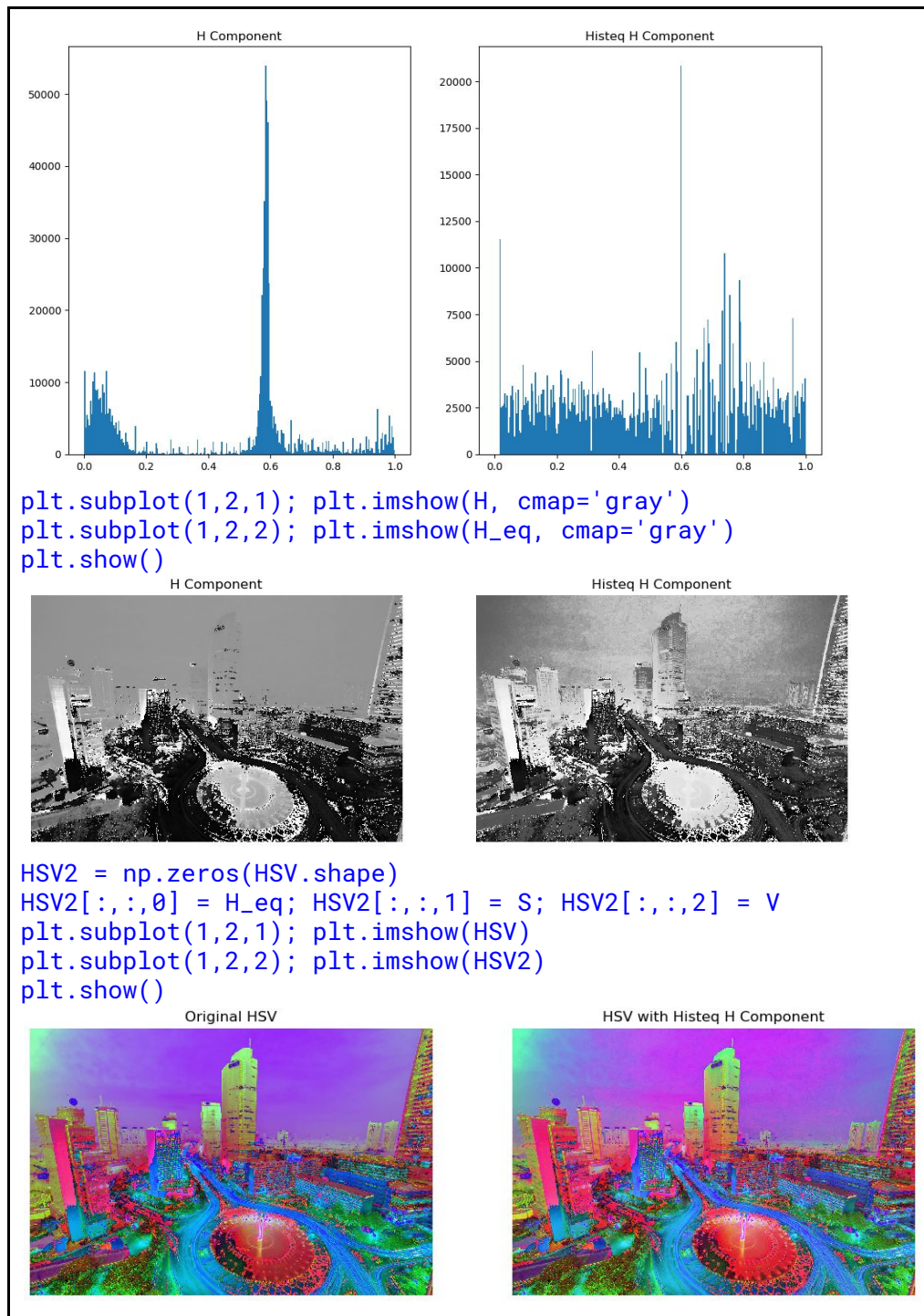
```
plt.subplot(1,3,1); plt.hist(C.flatten(), 256, range=(0,1))
plt.subplot(1,3,2); plt.hist(M.flatten(), 256, range=(0,1))
plt.subplot(1,3,3); plt.hist(Y.flatten(), 256, range=(0,1))
plt.show()
```

- Histogram Equalization

Histogram Equalization pada citra grayscale dan citra berwarna adalah sama, namun ekualisasi pada citra berwarna dilakukan secara terpisah pada setiap komponen warna atau pada salah satu komponen saja, sesuai kebutuhan. Contohnya, pada citra HSV dilakukan ekualisasi pada komponen H saja kemudian hasilnya digabungkan kembali dengan komponen lainnya.

```
H_eq = exposure.equalize_hist(H)
plt.subplot(1,2,1); plt.hist(H.flatten(),256,range=(0,1))
plt.subplot(1,2,2); plt.hist(H_eq.flatten(),256,range=(0,1))
plt.show()
```




- Smoothing


Smoothing pada citra berwarna dapat dilakukan pada setiap komponen warna atau pada satu komponen saja. Contohnya, dilakukan smoothing dengan average filter pada komponen R pada citra RGB.

```
R2 = filters.rank.mean(R, selem=morphology.square(9))
RGB2 = util.img_as_ubyte(np.zeros(RGB.shape))
RGB2[:, :, 0] = R2; RGB2[:, :, 1] = G; RGB2[:, :, 2] = B
```

Original RGB



RGB with Smoothed R Component

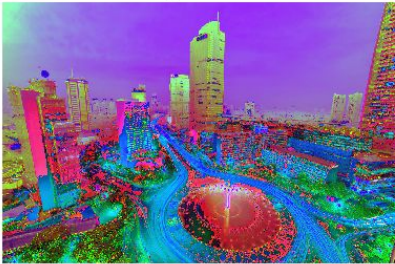


```

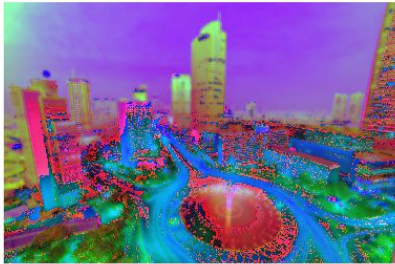
S2=util.img_as_float(filters.rank.mean(S,selem=morphology.square(9)))
HSV2 = np.zeros(HSV.shape)
HSV2[:, :, 0] = H; HSV2[:, :, 1] = S2; HSV2[:, :, 2] = V

```

Original HSV



HSV with Smoothed S Component




```


Res = color.hsv2rgb(HSV2)

```

Smoothed RGB



Smoothed HSV to RGB



- Sharpening


Sharpening pada citra berwarna dapat dilakukan pada setiap komponen warna maupun pada salah satu komponen saja. Berikut adalah contoh hasil sharpening yang dilakukan pada G pada citra RGB.

```


G2 = util.img_as_ubyte(filters.unsharp_mask(G, radius=5, amount=2))

```

Original G



Sharpened G Component



```

RGB2 = util.img_as_ubyte(np.zeros(RGB.shape))
RGB2[:, :, 0] = R; RGB2[:, :, 1] = G2; RGB2[:, :, 2] = B

```


Original RGB



RGB with Sharpened G Component

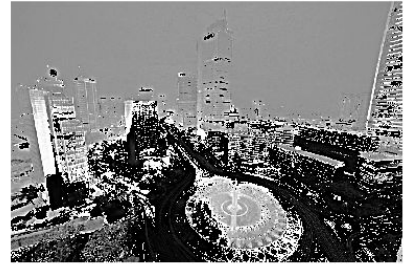


```
H2 = filters.unsharp_mask(H, radius=5, amount=2)
```

Original H

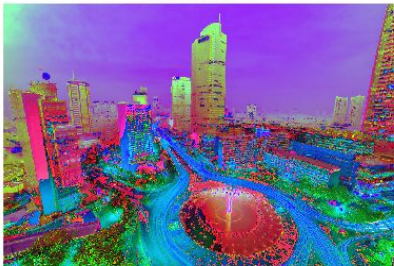


Sharpened H Component

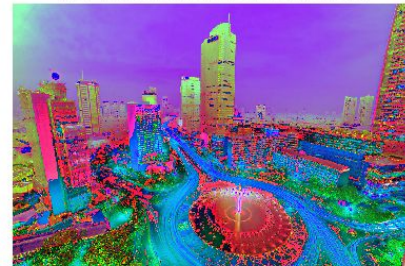


```
HSV2 = np.zeros(HSV.shape)  
HSV2[:, :, 0] = H2; HSV2[:, :, 1] = S; HSV2[:, :, 2] = V
```

Original HSV



HSV with Sharpened H Component



```
Res = color.hsv2rgb(HSV2)
```

Sharpened RGB



Sharpened HSV to RGB

