Lab 1
Pengolahan Citra
Image Enhancement in Spatial Domain
Rabu, 30 September 2020

---

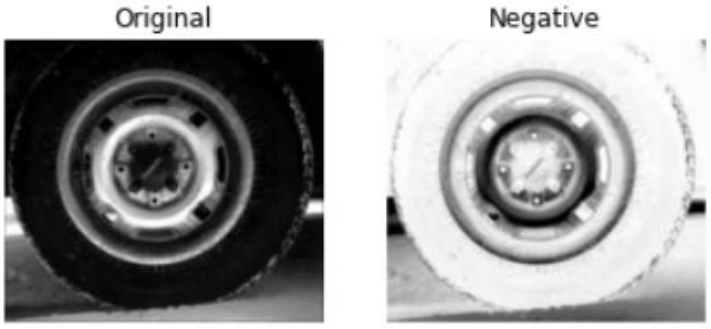
1. *Point Processing*

Point processing merupakan operasi pemrosesan citra yang melibatkan satu piksel saja. Selanjutnya akan dibahas beberapa teknik *point processing*.

A. *Image Negative*

$G_{baru} = 255 - G_{lama}$

```
i1 = io.imread('tire.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 Processing*

$s = c \cdot log(1 + r)$

s = citra yang dihasilkan oleh *log processing*
c = konstanta
r = citra yang akan ditransformasi

- Memetakan suatu citra dengan range warna sempit menjadi lebih lebar pada citra *output*-nya.
- Tujuannya untuk ekspansi nilai piksel gelap dan kompresi nilai piksel 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 Transformation

$s = cr^p$

s = citra hasil *gamma transformation*
c = konstanta
p = konstanta
r = citra masukan

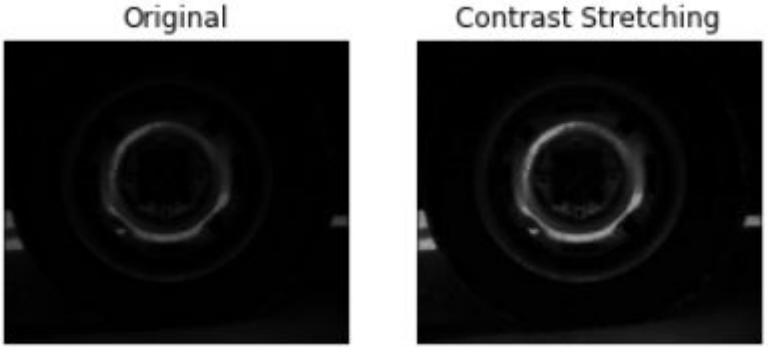
```
a = i1/255
c = 1
p1 = 0.5
p2 = 2
f1 = c*(a**p1)
f2 = c*(a**p2)
plt.subplot(1,3,1); plt.imshow(i1)
plt.title('Original'); plt.axis("off")
plt.subplot(1,3,2); plt.imshow(f1)
plt.title('P = 0.5'); plt.axis("off")
plt.subplot(1,3,3); plt.imshow(f2)
plt.title('P = 2'); plt.axis("off")
```

```
plt.show()
```

D. Contrast Stretching

- *Contrast stretching* mengembangkan range level intensitas piksel 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('dark_tire.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()
```

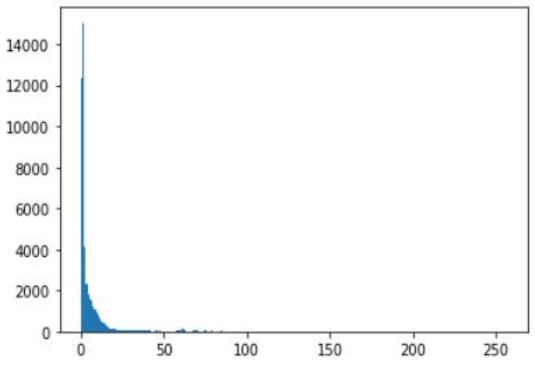


E. Image Histogram

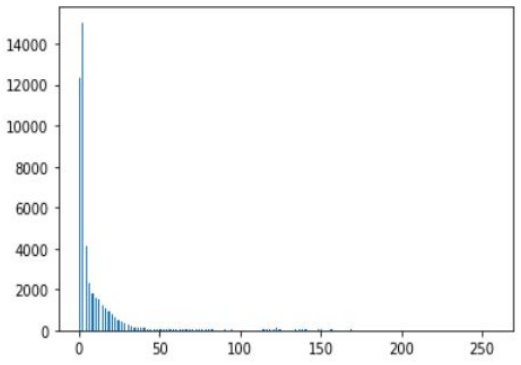
Histogram merupakan suatu teknik domain spasial yang mampu memberikan deskripsi global pada tampilan citra.

```
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()
```

Histogram (Original)



Histogram (Contrast stretching)

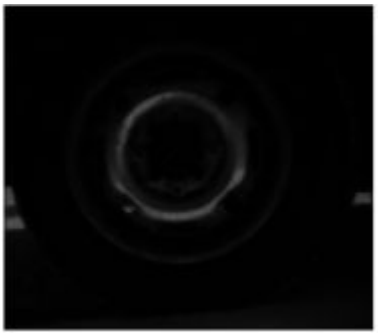


F. Histogram Equalization

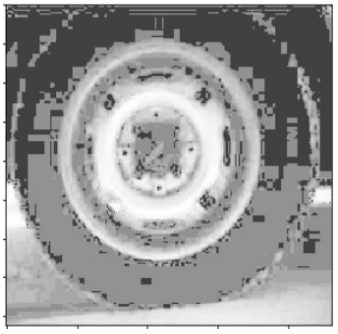
- Mengubah pemetaan *gray level* agar sebarannya (kontrasnya) lebih menyebar pada kisaran 0-255.
- *Histogram processing* digunakan untuk mengubah bentuk histogram agar pemetaan gray level pada citra juga berubah.

```
from skimage import exposure
eq_i3 = exposure.equalize_hist(i3)
```

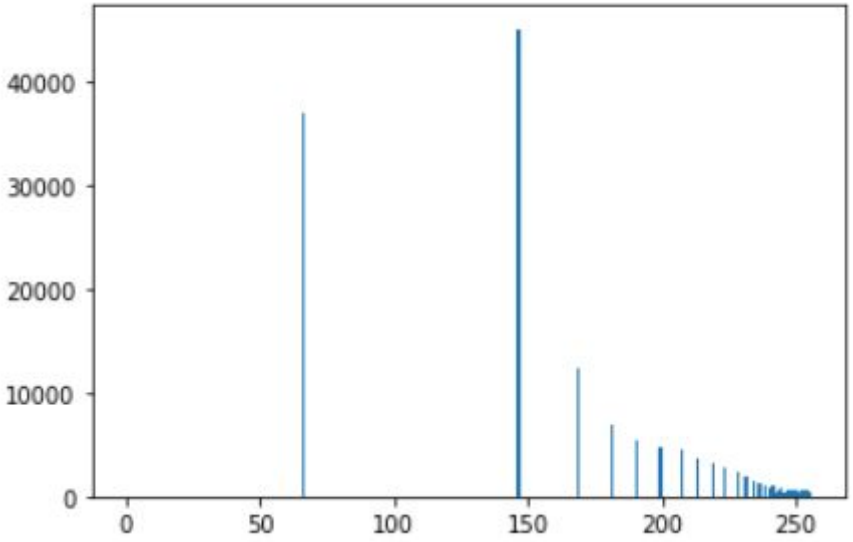
Original



Histogram equalization



Histogram-equalization



G. Image Substraction

Dilakukan jika kita ingin mengambil bagian tertentu saja dari citra.

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

image_1 image_2 image_1 - image_2

2. *Mask Processing*

- *Mask Processing* sering juga disebut dengan ***Image Filtering***.
- Operasi ini mempertimbangkan intensitas piksel dalam *neighborhood* dan koefisien filter yang berdimensi sama dengan *neighborhood*.
- Filter pada dasarnya adalah sebuah metode untuk meredam atau menghilangkan *noise* pada citra digital.
- Terdapat 2 jenis filter spasial; ***smoothing filter*** dan ***sharpening filter*.**

A. Smoothing Filter

- *Smoothing* digunakan untuk mengaburkan citra dan mereduksi *noise*.
- Biasa dilakukan saat *pre-processing* citra.
- Efek positif: Mereduksi *noise* dan detail yang tidak relevan dalam suatu citra.
- Efek negatif: Pengkaburan *edge*.
- Bisa dilakukan dengan menggunakan linear filter ataupun non linear filter.

1) Linear Filter

Metode yang digunakan: **korelasi** atau **konvolusi** (perkalian setiap piksel di lingkungan dengan koefisien yang sesuai dan menjumlahkan hasil untuk mendapatkan respon pada setiap titik (x, y).)

Berikut beberapa macam filter linear yang dibahas pada Lab ini:

- ***Average Filter***
  Mengganti intensitas tiap piksel dalam citra dengan rata-rata intensitas pada *neighborhood*.

```
from skimage import filters, morphology
gray = color.rgb2gray(io.imread('tire.jpg'))
fi = filters.rank.mean(gray,selem=morphology.square(4))
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()
```

Original

Average Filtered

- **Gaussian Filter**

  Mengganti intensitas tiap piksel dalam citra dengan hasil image filtering dari operasi konvolusi antara intensitas tiap piksel dalam citra dengan distribusi Gaussian.

  Contoh langkah-langkah diterapkannya Gaussian Filter pada suatu gambar berwarna:

  1. Gambar diperkecil hingga 5x5 piksel dan diekstraksi/direpresentasikan ke dalam matriks

$$\begin{bmatrix} 123 & 123 & 156 & 156 & 106 \\ 115 & 123 & 177 & 154 & 101 \\ 121 & 90 & 126 & 113 & 146 \\ 106 & 77 & 110 & 118 & 174 \\ 104 & 56 & 128 & 176 & 110 \end{bmatrix}$$

  2. Melakukan penentuan filter/mask yang ditentukan dengan distribusi Gaussian 2D. Contohnya menggunakan kernel berukuran 3x3.

$$\begin{bmatrix} 0.075 & 0.124 & 0.075 \\ 0.124 & 0.204 & 0.124 \\ 0.075 & 0.124 & 0.075 \end{bmatrix}$$

  3. Melakukan image filtering dengan menerapkan operasi konvolusi sebagai berikut:

     h(m,n) = (123 * 0.075) + (123 * 0.124) + (156 * 0.075) + (115 * 0.124) + (123 * 0.204) + (177 * 0.124) + (121 * 0.075) + (90 * 0.124) + (126 * 0.075)

sehingga menghasilkan:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 127 & 142 & 139 & 0 \\ 0 & 112 & 122 & 133 & 0 \\ 0 & 96 & 111 & 133 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Contoh penerapan kode Gaussian Filtering pada gambar lain:

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

Original

Gaussian Filtered (sigma = 1)

2) Non Linear Filter

- Responnya didasarkan pada pengurutan (ranking) dari intensitas piksel-piksel yang dilingkup oleh filter.
- Selanjutnya, intensitas piksel pada pusat filter diganti dengan intensitas hasil pengurutan.
- Filter non-linear memiliki lebih banyak keunggulan dibandingkan filter linear dengan ukuran filter yang sama.

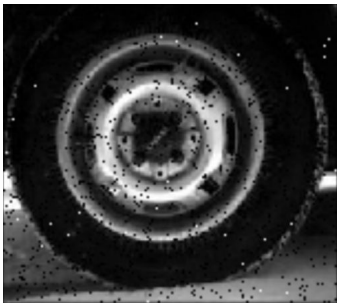
Berikut beberapa macam filter nonlinear yang dibahas pada Lab ini:

- **Median Filter**
  - Mengganti intensitas piksel pada pusat filter dengan median dari intensitas *neighborhood*.
  - Efektif untuk menghilangkan *impulse noise/salt-pepper noise.*
  - Intensitas piksel dalam neighborhood diurutkan → menentukan nilai median → mengganti intensitas piksel pada pusat *neighborhood* dengan median.

```
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(4))
```

Original

Noisy



Median Filtered

- **Minimum Filter**
  - Mengganti intensitas piksel pada pusat filter dengan minimum dari intensitas *neighborhood*.
  - Biasa digunakan untuk menghilangkan *positive outlier noise*.

```
fi = filters.rank.minimum(gray, selem=morphology.square(4))
```

Original

Minimum Filtered

- **Maximum Filter**
  - Mengganti intensitas piksel pada pusat filter dengan maksimum dari intensitas *neighborhood*.
  - Biasa digunakan untuk menghilangkan *negative outlier noise*.

```
fi = filters.rank.maximum(gray, selem=morphology.square(4))
```

Original

Maximum Filtered

B. Sharpening Filter

- *Sharpening* digunakan untuk menonjolkan detail dalam citra atau untuk mempertajam detail dalam citra atau untuk mempertajam detail yang kabur.
- Biasa dilakukan saat *pre-processing* citra.
- Kebalikan dari *smoothing* yang dilakukan dengan rata-rata (dengan integrasi), *sharpening* dapat dilakukan dengan **derivatif** (penurunan).

- **Robert, Prewitt, Sobel *(Edge Detection)***
  Edge detection menghasilkan tepi-tepi dari objek citra. Suatu titik (x, y) dikatakan sebagai tepi (*edge*) dari suatu citra apabila mempunyai perbedaan yang tinggi dengan tetangganya.
  - **Roberts**: Teknik diferensial pada arah horizontal dan diferensial pada arah vertikal, dengan ditambahkan proses konvensi biner setelah dilakukan diferensial. Kernel filter yang digunakan dalam metode Roberts:

$$H = \begin{bmatrix} -1 & 1 \end{bmatrix} \text{dan } H = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

  - **Prewitt**: Pengembangan metode Robert dengan menggunakan filter HPF *(High Pass Filter)* yang diberi satu angka nol penyangga. Metode ini mengambil prinsip dari fungsi laplacian yang dikenal sebagai fungsi untuk membangkitkan HPF (*High Pass Filter*). Kernel *filter* yang digunakan dalam metode Prewitt:

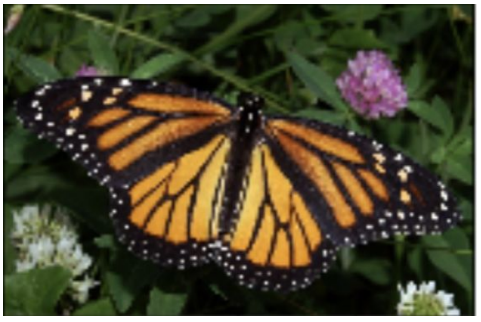
$$H = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \text{dan } H = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

- **Sobel**: Pengembangan metode Sobel dengan menggunakan filter HPF *(High Pass Filter)* yang diberi satu angka nol penyangga. Kelebihannya adalah kemampuan untuk mengurangi *noise* sebelum melakukan perhitungan deteksi tepi. Kernel *filter* yang digunakan dalam metode Sobel:

$$H = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{dan } H = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

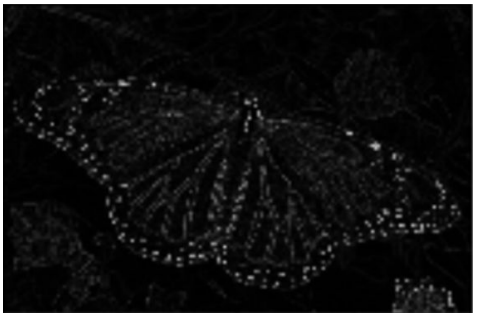
```
gray = color.rgb2gray(i2)
ed1 = util.img_as_ubyte(filters.prewitt(gray))
ed2 = util.img_as_ubyte(filters.roberts(gray))
ed3 = util.img_as_ubyte(filters.sobel(gray))
```

Original



Prewitt Filtered

Roberts Filtered



Sobel Filtered

```
#Sharpening Image
```

```
gray = color.rgb2gray(i2)
fi = filters.rank.mean(gray, selem=morphology.square(3))
sh = filters.unsharp_mask(fi, radius=3)
```

Original

Blurred

Sharpened