

Lab 3

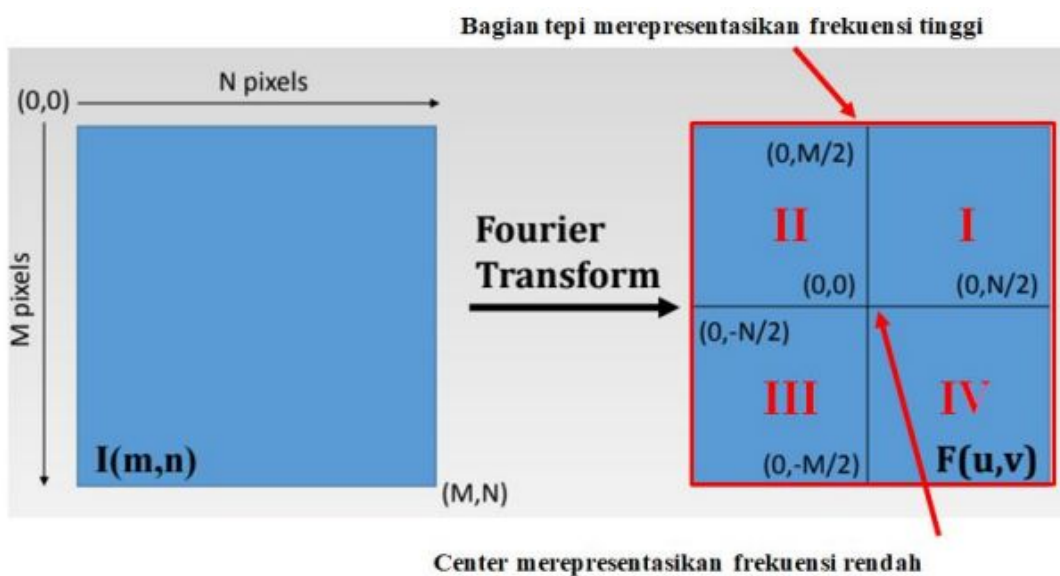
Pengolahan Citra

Image Enhancement In the Frequency Domain

Jumat, 11 Oktober 2019

A. Image Enhancement In the Frequency Domain

Salah satu transformasi yang dapat digunakan untuk mengetahui informasi frekuensi pada sebuah citra adalah transformasi Fourier. Sebuah citra adalah masukan yang memiliki 2 dimensi sehingga dapat ditransformasi menggunakan 2D-Discrete Fourier Transform (2D-DFT). Salah satu algoritma yang dapat digunakan untuk menghitung DFT menggunakan Python adalah Fast Fourier Transform (FFT).

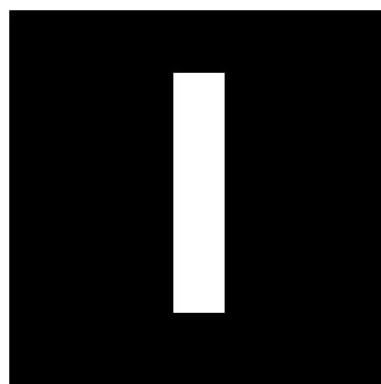


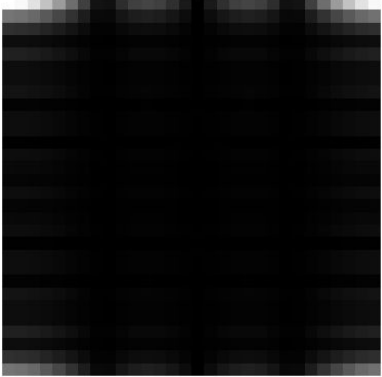
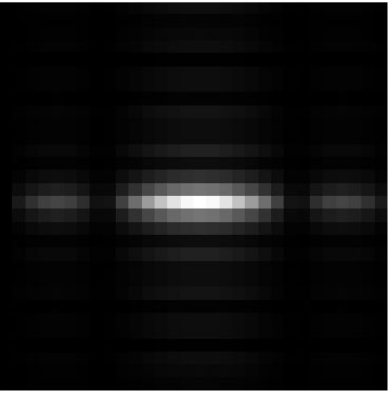
- Menampilkan Fourier Spectrum

```
# Import Library yang dibutuhkan.
import numpy as np
from scipy import fftpack as fp
import matplotlib.pyplot as plt

# Misalnya kita membuat sebuah gambar
persegi panjang

f = np.zeros((30,30))
f[5:24,13:17] = 1
plt.imshow(f, cmap='gray'); plt.show()
```



| | |
|---|---|
| <pre># Menghitung DFT, Ft terdiri dari real dan imaginary. Ft = fp.fft2(f); # abs merupakan fungsi untuk menghitung magnitudo kombinasi dari kedua komponen tersebut. f2 = abs(Ft) # Menampilkan spectrum fourier dari gambar plt.imshow(f2, cmap='gray'); plt.show()</pre> |  |
| <pre># Koefisien frekuensi-nol ditampilkan di sudut kiri atas. Untuk menampilkannya di tengah, gunakan fungsi fftshift. f2 = fp.fftshift(f2) f2 = abs(f2) plt.imshow(f2, cmap='gray'); plt.show()</pre> |  |

- Basic Step in DFT Filtering (Gonzalez, 2003 : 121)

1. Obtain the padding parameters using function paddedsize:
fungsi paddedsize terdapat di file helper.py
`w, l = paddedsize(i1.shape)`
2. Obtain the Fourier transform of the image with padding:
`F = fp.fft2(i1,(w,l))`
3. Generate a filter function, h, the same size as the image
4. Multiply the transformed image by the filter:
`G = h*f`
5. Obtain the real part of the inverse FFT of G:
`res = fp.ifft2(G).real`
6. Crop the top, left rectangle to the original size:
`res = res[:i1.shape[0], :i1.shape[1]]`

- Contoh: Menggunakan Filter Sobel pada domain Frekuensi

| Spatial Domain Filtering | Frequency Domain Filtering |
|--|---|
| <pre># Membuat filter spasial from scipy import signal from skimage import io, color, util f = io.imread('lena.bmp') f = color.rgb2gray(f)</pre> | <pre>from skimage import io, color, util f = io.imread('lena.bmp') f = color.rgb2gray(f); h = np.array([[1,2,1], [0,0,0], [-1,-2,-1]]) # sobel operator</pre> |

```
h = np.array([[1,2,1], [0,0,0],
[-1,-2,-1]]) # sobel operator
sf = signal.convolve2d(f,h)
plt.imshow(sf, cmap='gray')
plt.axis("off"); plt.show()
```



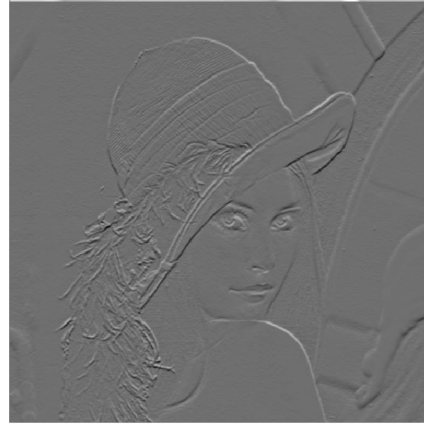
```
#Gunakan fungsi abs untuk
memperoleh magnitude yg sesuai
sfi = abs(sf)
plt.imshow(sfi, cmap='gray')
plt.axis("off"); plt.show()
```



```
# Threshold untuk memperoleh citra
biner
bin = sfi > 0.2*max(sfi.flatten())
plt.imshow(bin, cmap='gray')
plt.axis("off"); plt.show()
```



```
w,l = paddedsize(f.shape[0],
f.shape[1])
f2 = fp.fft2(f,(w,l))
h2 = fp.fft2(h,(w,l))
F_fh = h2*f2
res = fp.ifft2(F_fh).real
res = res[:f.shape[0],:f.shape[1]]
plt.imshow(res, cmap='gray')
plt.axis("off"); plt.show()
```

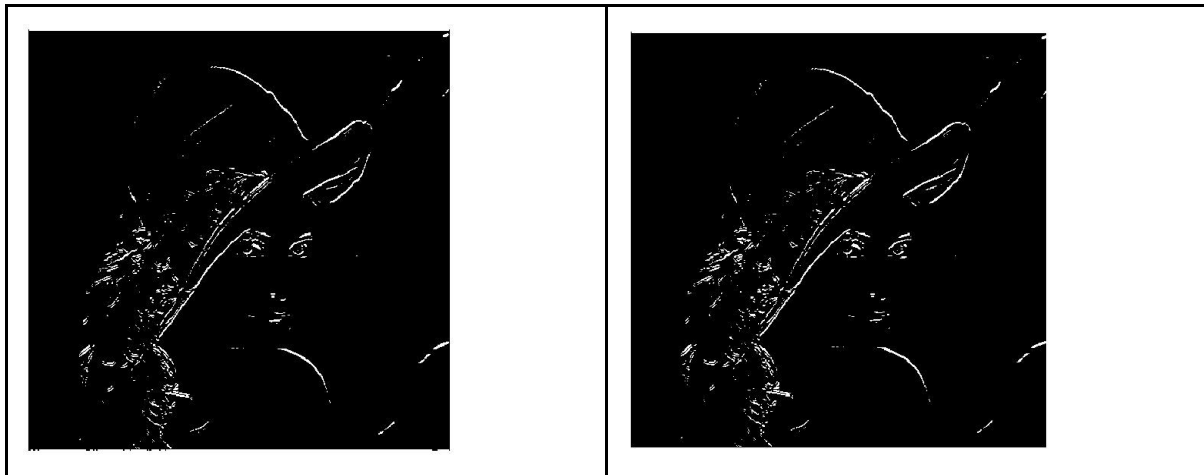


```
# Gunakan fungsi abs untuk
memperoleh magnitude yg sesuai
ffi = abs(res)
plt.imshow(ffi, cmap='gray')
plt.axis("off"); plt.show()
```



```
# Threshold untuk memperoleh citra
biner
bin = ffi > 0.2*max(ffi.flatten())
plt.imshow(bin, cmap='gray')
plt.axis("off"); plt.show()
```





- **Menggunakan Spesifik Filter pada domain Frekuensi (Lowpass Filter)**

- Lowpass filter membuat citra menjadi blurring/smoothing
- Bekerja dengan cara mengurangi (suppress) nilai Fourier Transform pada frekuensi tinggi dan meloloskan frekuensi rendah
- Ada 3 jenis lowpass filter yang umum digunakan yaitu:

| Ideal Lowpass Filter (ILPF) | Butterworth Lowpass Filter (BLPF) | Gaussian Lowpass Filter (GLPF) |
|------------------------------------|--|---------------------------------------|
| | | |

Contoh: Dilakukan Gaussian Lowpass filtering (GLPF) pada citra lena:

```
i1 = color.rgb2gray(io.imread('lena.bmp'))
w, l = paddedsize(i1.shape[0], i1.shape[1])

# membuat gaussian lowpass filter
# fungsi lpfilter terdapat di file helper.py
h = lpfilter('gaussian', w, l, 0.05 * w)

# menghitung DFT citra
f = fp.fft2(i1,(w,l))

# Apply lowpass filter
LPFS_lena = h*f

# convert ke domain spasial
LPF_lena = fp.ifft2(LPFS_lena).real
LPF_lena = LPF_lena[:i1.shape[0], :i1.shape[1]]
```

```
# Menampilkan fourier spectrum
Fc = fp.fftshift(f)
Fcf = fp.fftshift(LPFS_lena)

# fungsi abs untuk menghitung magnitude
S1 = np.log(1+abs(Fc))
S2 = np.log(1+abs(Fcf))

plt.subplot(1,2,1); plt.imshow(i1, cmap='gray')
plt.title("Original"); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(S1, cmap='gray')
plt.title("Original Fourier Spectrum"); plt.axis("off")
plt.show()
```


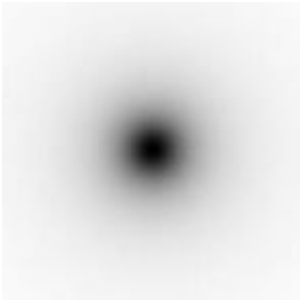



```
plt.subplot(1,2,1); plt.imshow(LPF_lena, cmap='gray')
plt.title("Gaussian Lowpass Filter"); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(S2, cmap='gray')
plt.title("Spectrum of Filtered Image"); plt.axis("off")
plt.show()
```



- Menggunakan Spesifik Filter pada domain Frekuensi (Highpass Filter)

- Lowpass filter membuat tepi citra menjadi tajam/sharpening
- Bekerja dengan cara mengurangi (suppress) nilai Fourier Transform pada frekuensi rendah dan meloloskan frekuensi tinggi.
- Ada 3 jenis highpass filter yang umum digunakan yaitu:

| Ideal Highpass Filter (IHPF) | Butterworth Highpass Filter (BHPF) | Gaussian Highpass Filter (GHPF) |
|---|---|---|
|  |  |  |

Contoh: Dilakukan Gaussian Highpass filtering (GLPF) pada citra lena:

```

i1 = color.rgb2gray(io.imread('lena.bmp'))
w, l = paddedsize(i1.shape[0], i1.shape[1])

# membuat gaussian highpass filter
# fungsi hpfilter terdapat di file helper.py
h = hpfilter('gaussian', w, l, 0.05 * w)

# menghitung DFT citra
f = fp.fft2(i1,(w,l))

# Apply highpass filter
LPFS_lena = h*f

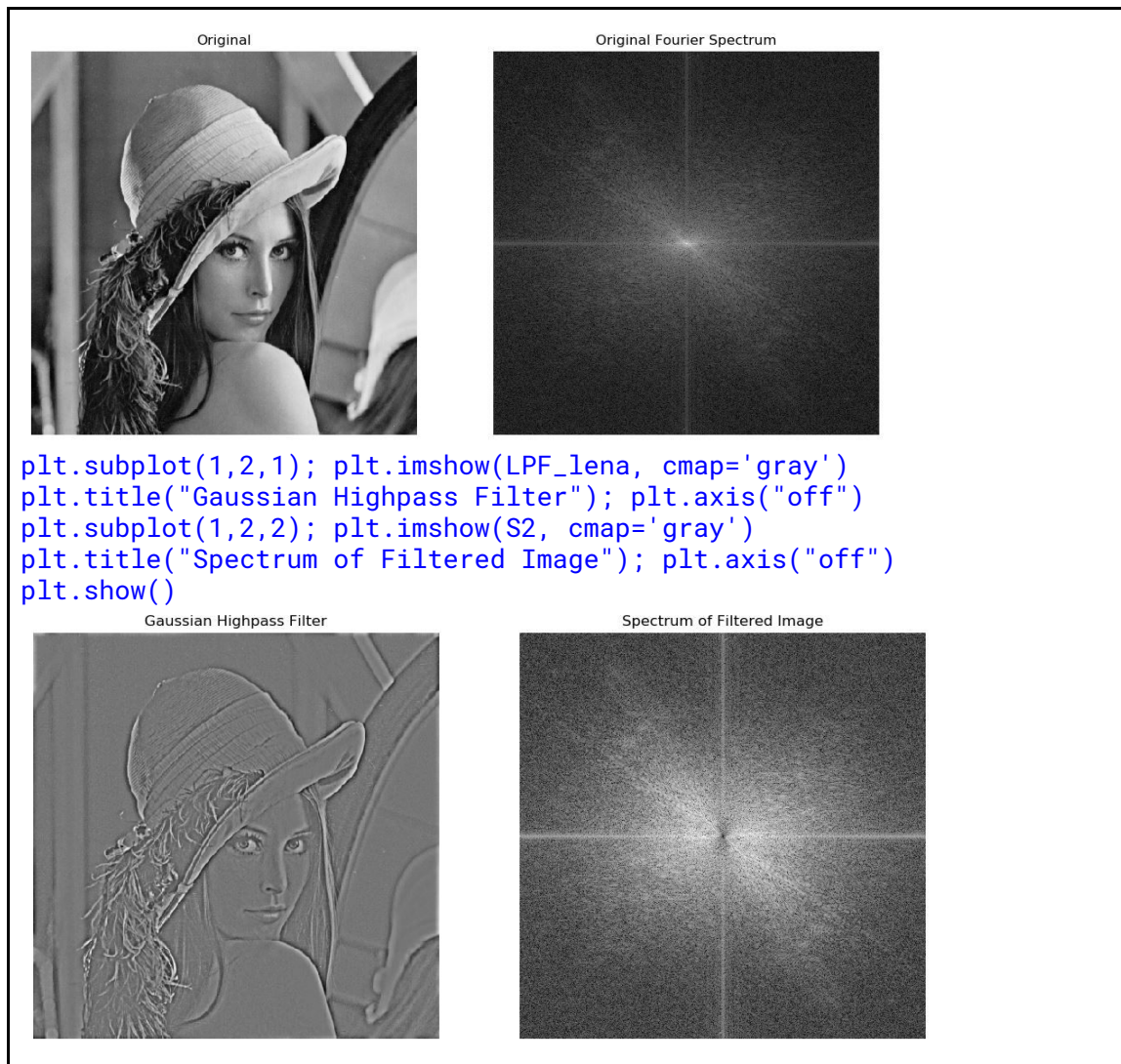
# convert ke domain spasial
LPF_lena = fp.ifft2(LPFS_lena).real
LPF_lena = LPF_lena[:i1.shape[0], :i1.shape[1]]

# Menampilkan fourier spectrum
Fc = fp.fftshift(f)
Fcf = fp.fftshift(LPFS_lena)

# fungsi abs untuk menghitung magnitude
S1 = np.log(1+abs(Fc))
S2 = np.log(1+abs(Fcf))

plt.subplot(1,2,1); plt.imshow(i1, cmap='gray')
plt.title("Original"); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(S1, cmap='gray')
plt.title("Original Fourier Spectrum"); plt.axis("off")
plt.show()

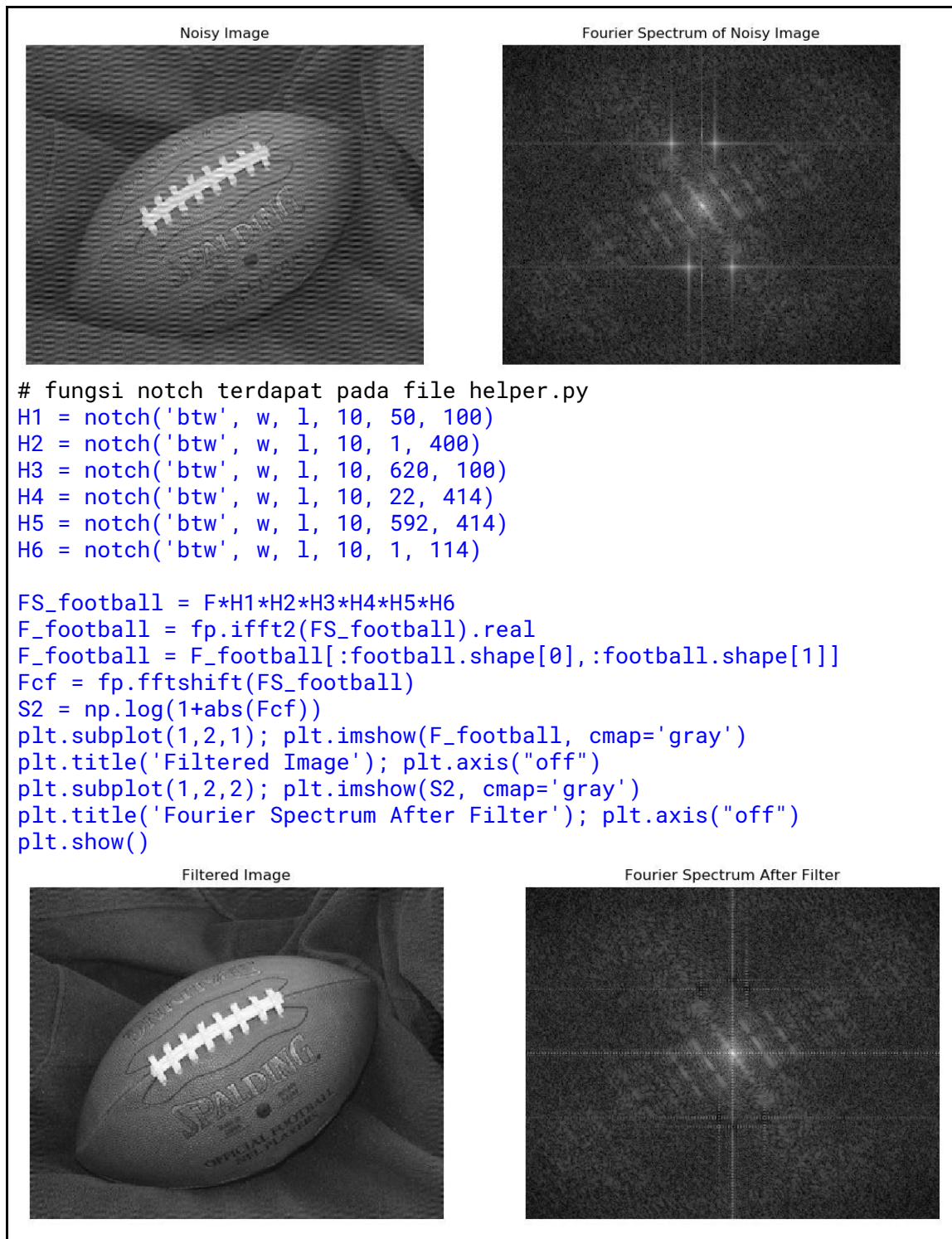
```

- Notch Filter

- Digunakan untuk menghapus noise "Spectral" berulang dari sebuah gambar.
- Mengurangi frekuensi yang dipilih (dan beberapa tetangganya) dan meloloskan frekuensi yang lain.

```
football = io.imread('noiseball.png')
w, l = paddedsize(football.shape[0], football.shape[1])
F = fp.fft2(util.img_as_float(football), (w, l))
Fc = fp.fftshift(F)
S1 = np.log(1+abs(Fc))
plt.subplot(1,2,1); plt.imshow(football, cmap='gray')
plt.title('Noisy Image'); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(S1, cmap='gray')
plt.title('Fourier Spectrum of Noisy Image'); plt.axis("off")
plt.show()
```



B. Morphological Image Processing

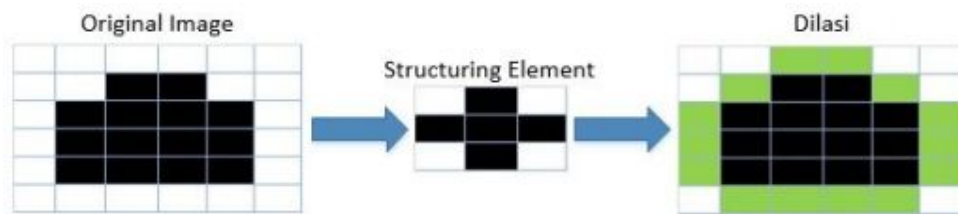
- Dilasi dan Erosi

Dilasi

Dilasi adalah operasi morfologi yang akan menambahkan pixel pada batas antar objek dalam suatu citra digital. Atau secara rinci Dilasi merupakan suatu proses menambahkan piksel pada batasan dari objek dalam suatu image sehingga nantinya

apabila dilakukan operasi ini maka citra tersebut hasilnya lebih besar ukurannya dibandingkan dengan citra aslinya.

Ilustrasi:



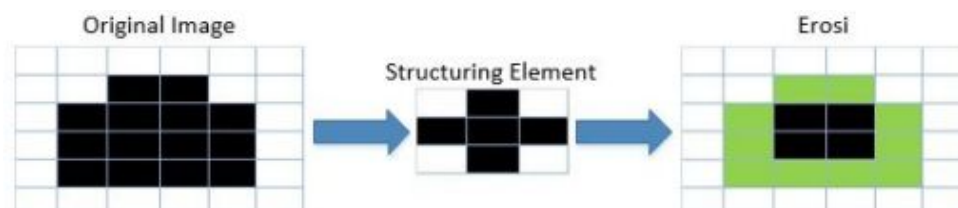
```
from skimage import morphology
BW = color.rgb2gray(io.imread('text.png'))
se = morphology.rectangle(1, 11)
BW2 = morphology.dilation(BW, se)
plt.subplot(1,2,1); plt.imshow(BW, cmap='gray')
plt.title('Original'); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(BW2, cmap='gray')
plt.title("Dilated"); plt.axis("off")
plt.show()
```



Erosi

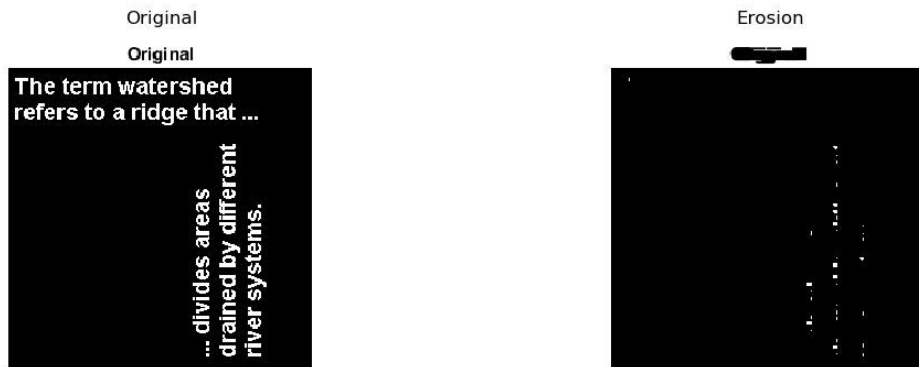
Erosi merupakan kebalikan dari Dilasi. Proses ini akan membuat ukuran sebuah citra menjadi lebih kecil. Berbeda dengan dilatasi, apabila erosi dilakukan maka yang dikerjakan adalah memindahkan piksel pada batasan-batasan objek yang akan di erosi. Jumlah dari piksel yang ditambah atau dihilangkan bergantung pada ukuran dan bentuk dari structuring element yang digunakan untuk memproses image tersebut.

Ilustrasi:



```
BW = color.rgb2gray(io.imread('text.png'))
se = morphology.rectangle(1, 11)
BW3 = morphology.erosion(BW, se)
plt.subplot(1,2,1); plt.imshow(BW, cmap='gray')
```

```
plt.title('Original'); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(BW3, cmap='gray')
plt.title("Erosion"); plt.axis("off")
plt.show()
```

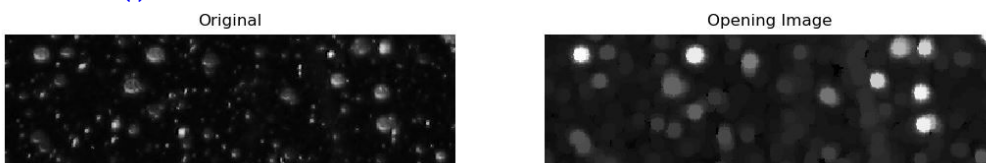


- Opening and Closing

Opening

Opening merupakan kombinasi proses dimana suatu citra digital dikenai operasi erosi dilanjutkan dengan dilasi. Operasi opening pada citra mempunyai efek memperhalus batas-batas objek, memisahkan objek-objek yang sebelumnya bergandengan, dan menghilangkan objek-objek yang lebih kecil daripada ukuran structuring.

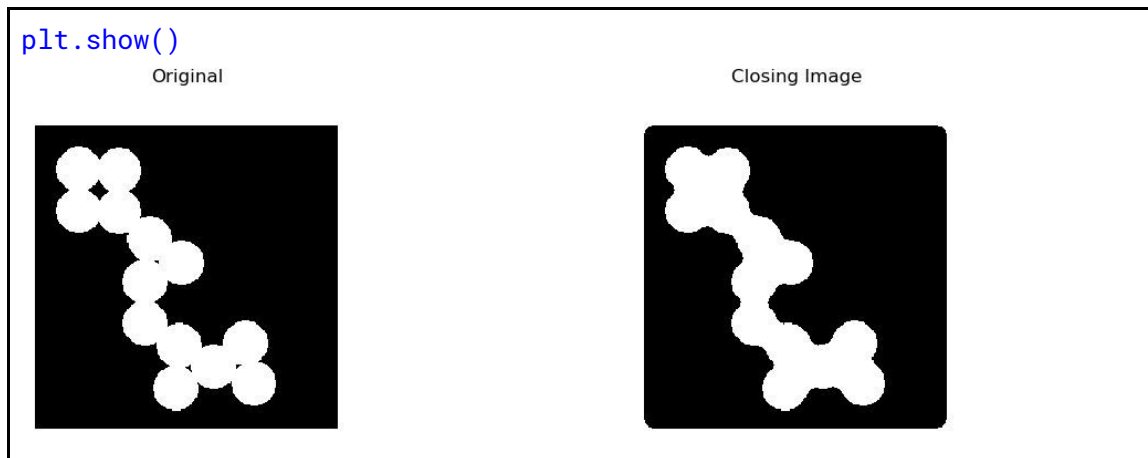
```
BW = color.rgb2gray(io.imread('snowflakes.png'))
se = morphology.disk(5)
BW2 = morphology.opening(BW, se)
plt.subplot(1,2,1); plt.imshow(BW, cmap='gray')
plt.title('Original'); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(BW2, cmap='gray')
plt.title("Opening Image"); plt.axis("off")
plt.show()
```



Closing

Closing merupakan kombinasi dimana suatu citra dikenai operasi dilasi dilanjutkan dengan erosi. Operasi closing juga cenderung akan memperhalus objek pada citra, namun dengan cara menyambung pecahan-pecahan dan menghilangkan lubang-lubang kecil pada objek.

```
BW = color.rgb2gray(io.imread('circle.png'))
se = morphology.disk(7)
BW2 = morphology.closing(BW, se)
plt.subplot(1,2,1); plt.imshow(BW, cmap='gray')
plt.title('Original'); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(BW2, cmap='gray')
plt.title("Closing Image"); plt.axis("off")
```



- Top and Bottom hat Filtering

Top Hat Filtering

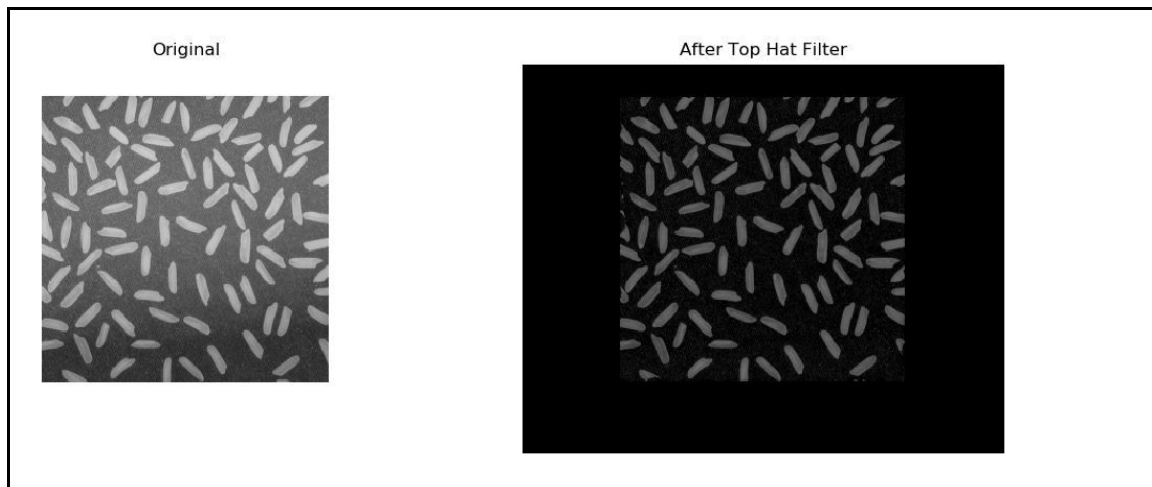
Computes morphological opening of the image.

SE (Structuring Element) is a single structuring element object returned by the some functions in morphology library.

- Flat morphological structuring element, binary value.

```
SE = morphology.diamond(4)
print(SE)
[[0 0 0 0 1 0 0 0 0]
 [0 0 0 1 1 1 0 0 0]
 [0 0 1 1 1 1 1 0 0]
 [0 1 1 1 1 1 1 1 0]
 [1 1 1 1 1 1 1 1 1]
 [0 1 1 1 1 1 1 1 0]
 [0 0 1 1 1 1 1 0 0]
 [0 0 0 1 1 1 0 0 0]
 [0 0 0 0 1 0 0 0 0]]
```

```
i1 = color.rgb2gray(io.imread('rice.png'))
se = morphology.disk(12)
i2 = morphology.white_tophat(i1, se)
plt.subplot(1,2,1); plt.imshow(i1, cmap='gray', vmin=0, vmax=1)
plt.title('Original'); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(i2, cmap='gray', vmin=0, vmax=1)
plt.title("After Top Hat Filter"); plt.axis("off")
plt.show()
```



Bottom Hat Filtering

Computes morphological closing of the image.

SE (Structuring Element) is a single structuring element object returned by the strel or offsetstrel functions. In scikit-image, we use black top hat for bottom hat filtering.

Contoh: contrast enhancement using top hat and bottom hat filtering

```
i1 = color.rgb2gray(io.imread('pout.png'))
se = morphology.disk(3)
i2 = np.subtract(np.add(i1, morphology.white_tophat(i1, se)),
morphology.black_tophat(i1, se))
plt.subplot(1,2,1); plt.imshow(i1, cmap='gray', vmin=0, vmax=1)
plt.title('Original'); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(i2, cmap='gray', vmin=0, vmax=1)
plt.title("Result After Filter"); plt.axis("off")
plt.show()
```

