Dasar-Image

May 6, 2021

0.1 ## Pengenalan Image Processing dengan Python

0.1.1 1. Import module yang diperlukan

Bagian ini memuat beberapa module wajib yang digunakan dalam notebook ini: **numpy**, **pandas**, **cv2**, **skimage**, **PIL**, **matplotlib**

- Numpy adalah library untuk manipulasi array, digunakan untuk aljabar linier, transformasi Fourier, dan kemampuan bilangan acak.
- Pandas adalah library untuk manipulasi data dan analisis data.
- CV2 adalah library untuk tugas-tugas computer vision.
- Skimage adalah library yang mendukung aplikasi pengolah gambar pada python.
- Matplotlib adalah library yang menghasilkan gambar dan menyediakan toolkit antarmuka pengguna grafis.

[1]:
```
import numpy as np
import pandas as pd
import cv2 as cv
from skimage import io
from PIL import Image
import matplotlib.pylab as plt
```

0.1.2 2. Baca Gambar dari Url

Pada langkah ini kita akan membaca gambar dari url, dan menampilkannya menggunakan openCV, perhatikan perbedaannya saat membaca gambar dalam format RGB dan BGR. Saluran warna input default dalam format BGR untuk openCV.

[7]:
```
# Buat daftar untuk menyimpan url gambar
urls = ["image-1.png",
        "image-2.png",
        "image-3.png"
      ]

# Fungsi untuk Baca dan tampilkan gambar
def cv_imshow(img):
    plt.figure(figsize = (15,15))
    plt.imshow(img)
    plt.xticks([]), plt.yticks([])
    plt.show()
```

```
# loop terhadap URL gambar, Anda dapat menyimpan beberapa url gambar dalam
 ↪daftar

for url in urls:
    image = io.imread(url)
    image_2 = cv.cvtColor(image, cv.COLOR_BGR2RGB)
    final_frame = cv.hconcat((image, image_2))
    cv_imshow(final_frame)
```

0.1.3 Tugas 1: Baca gambar dari URL dan tampilkan

Cari gambar dari google, lalu gunakan url address dari gambar tersebut untuk melakukan operasi di bawah ini dengan menghapus tanda komentarnya.

[9]:
```
# TODO: LOAD IMAGE FROM URL
url = 'image-3.png'
myImg = io.imread(url)
cv_imshow(cv.cvtColor(myImg, cv.COLOR_BGR2RGB))
```

0.1.4 3. Kontur Gambar dan Histogram

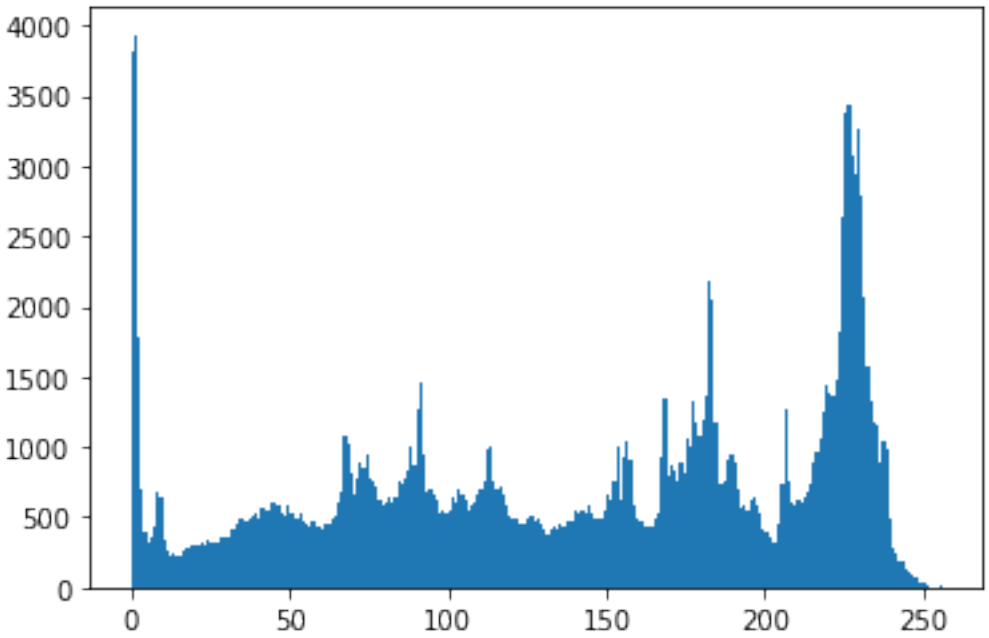
[10]:
```
# Gunakan image dari indeks pertama dalam variabel urls
image = io.imread(urls[0])
```

0.1.5 Menghasilkan Histogram citra berwarna dan citra grayscale

Terkadang Anda ingin meningkatkan kontras pada gambar atau memperluas kontras di wilayah tertentu sambil mengorbankan detail dalam warna yang tidak terlalu bervariasi, atau tidak penting. Alat yang baik untuk menemukan wilayah yang menarik adalah histogram. Untuk membuat histogram dari data gambar kita, kita menggunakan fungsi matplot.pylab hist ().

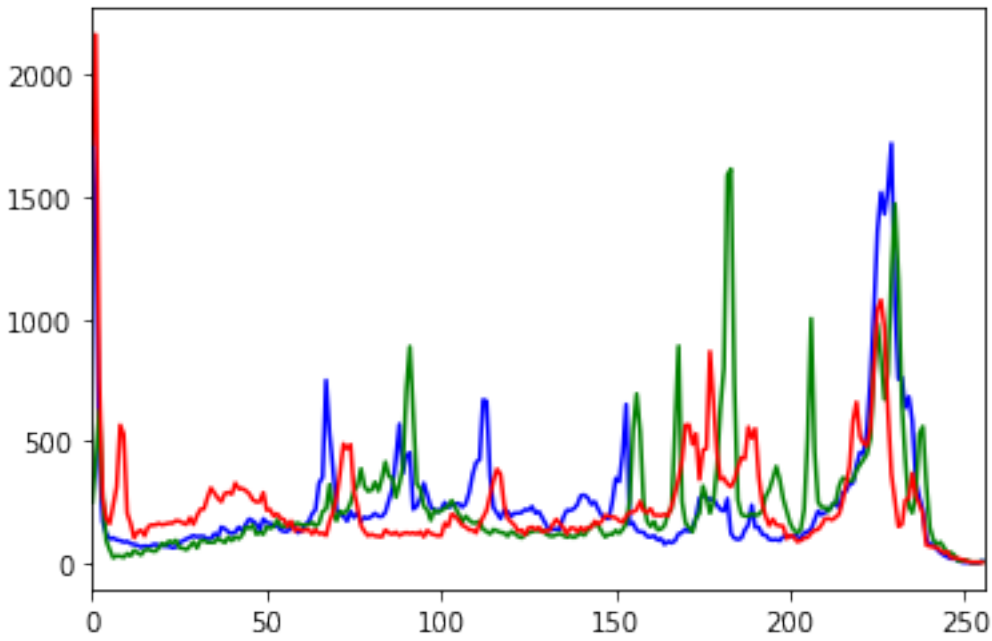
Menampilkan histogram dari semua piksel pada gambar berwarna :

[11]:
```
plt.hist(image.ravel(),bins = 256, range = [0,256])
plt.show()
```

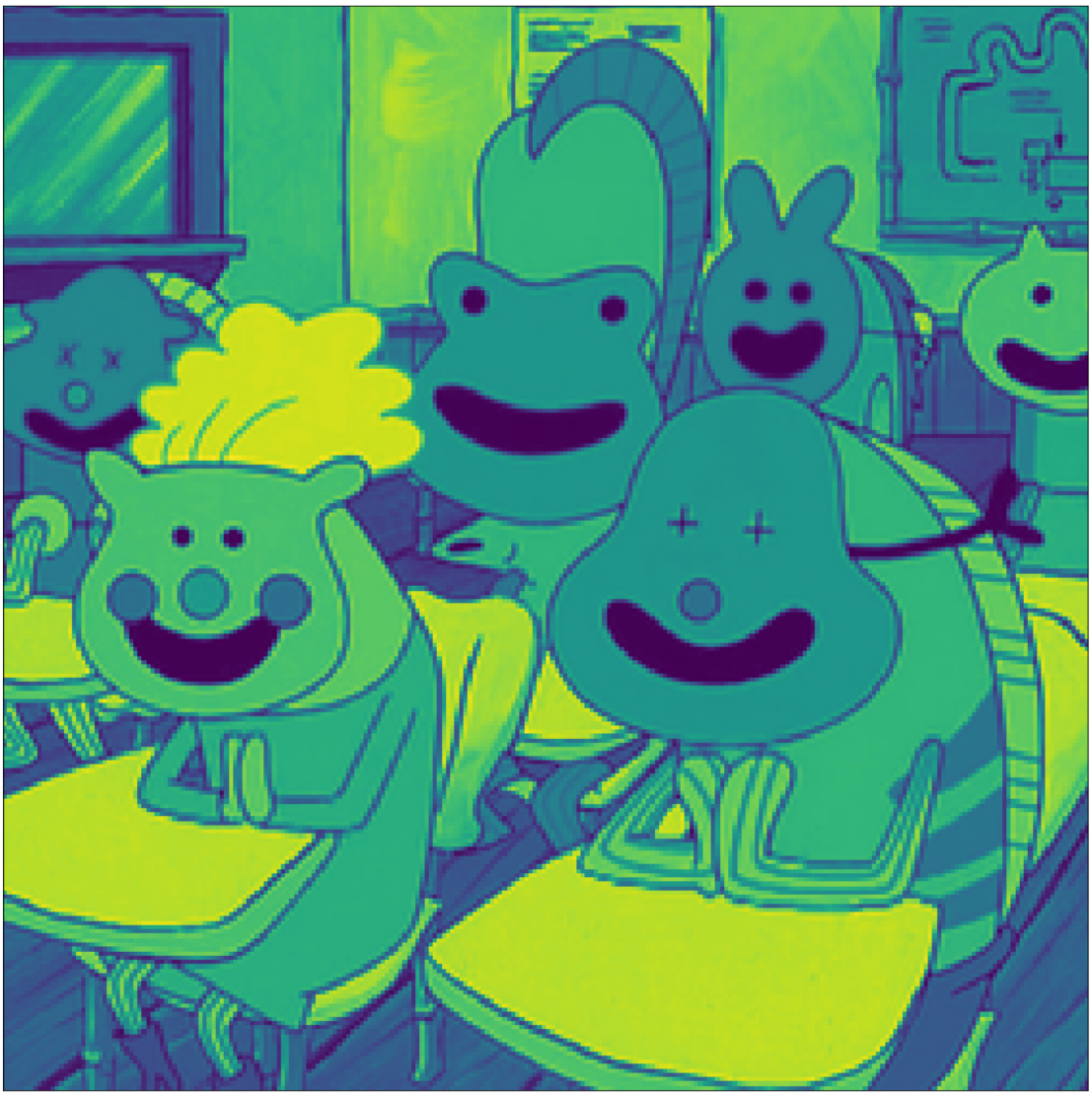


Menampilkan histogram saluran R, G, B Kita dapat mengamati bahwa saluran hijau memiliki banyak piksel di 255, yang mewakili tambalan putih pada gambar.

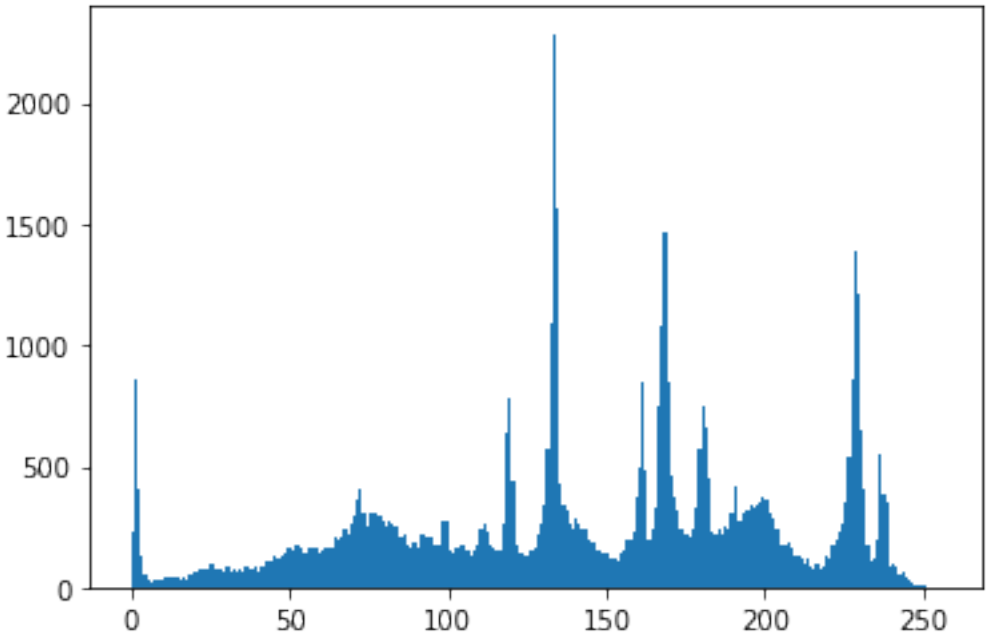
[12]:
```
color = ('b','g','r')
for i,col in enumerate(color):
    histr = cv.calcHist([image],[i],None,[256],[0,256])
    plt.plot(histr,color = col)
    plt.xlim([0,256])
plt.show()
```



[13]:
```
gray_image = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
cv_imshow(gray_image)
```



[14]:
```
# Plot histogram gambar abu-abu.
# Kita bisa mengamati bahwa frekuensi histori citra mengalami penurunan ~ 1/3␣
↪dari histogram citra berwarna
plt.hist(gray_image.ravel(),bins = 256, range = [0, 256])
plt.show()
```



0.1.6 Tugas 2: Tampilkan gambar anda dalam grayscale dan buat histogramnya

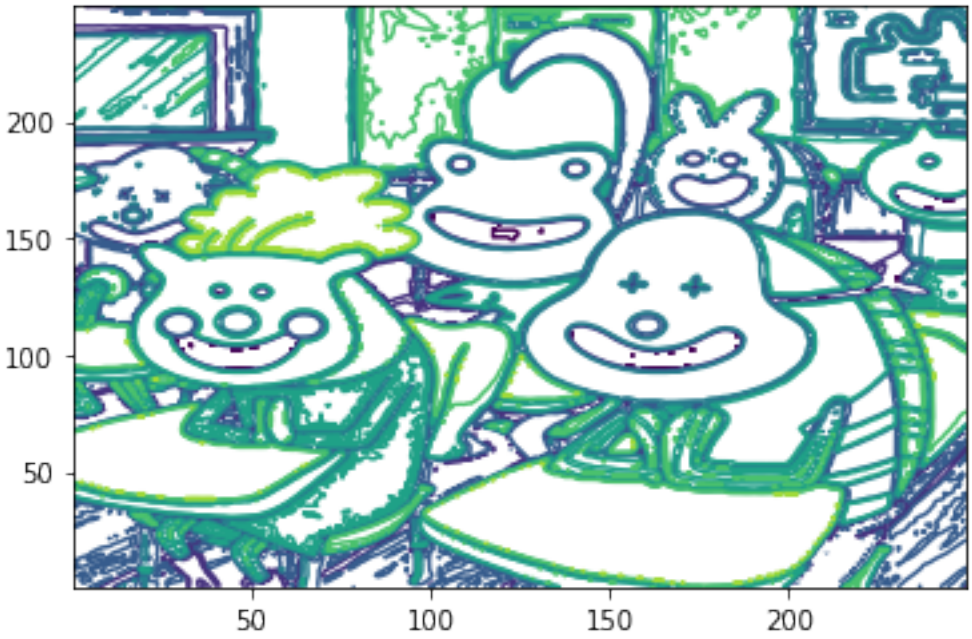
[16]:
```
myGrayImg = cv.cvtColor(myImg, cv.COLOR_BGR2GRAY)
cv_imshow(myGrayImg)
```

0.1.7 Temukan kontur gambar dari gambar grayscale

Metode 1: Gunakan matplotlib. contour

[17]:
```
plt.contour(gray_image, origin = "image")
```

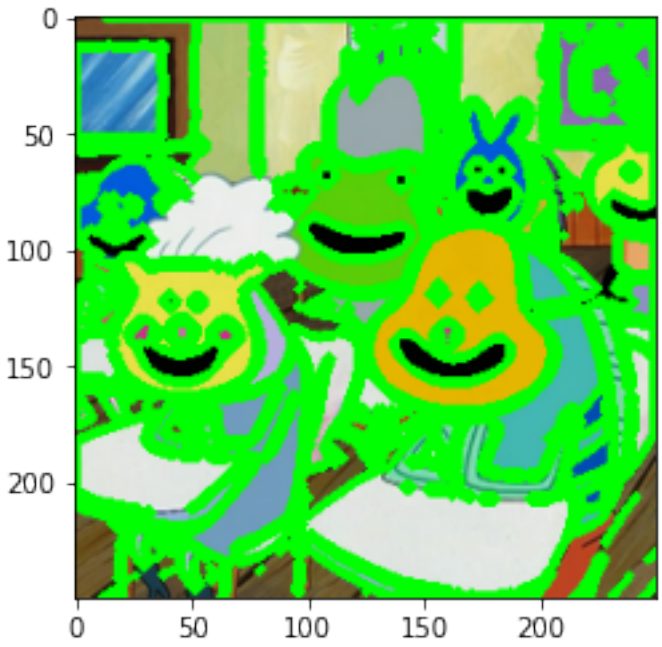
[17]:
```
<matplotlib.contour.QuadContourSet at 0x7efff04ee7c0>
```



Metode 2: Gunakan library openCV

[18]:
```
# Setel ambang batas untuk deteksi kontur
ret, thresh = cv.threshold(gray_image,100,200,0)
contours, hierarchy = cv.findContours(thresh, cv.RETR_TREE, cv.
 ↪CHAIN_APPROX_SIMPLE)
cv.drawContours(image, contours, -1, (0, 255, 0), 3)
plt.imshow(image)
```

[18]:
```
<matplotlib.image.AxesImage at 0x7efff0918d00>
```



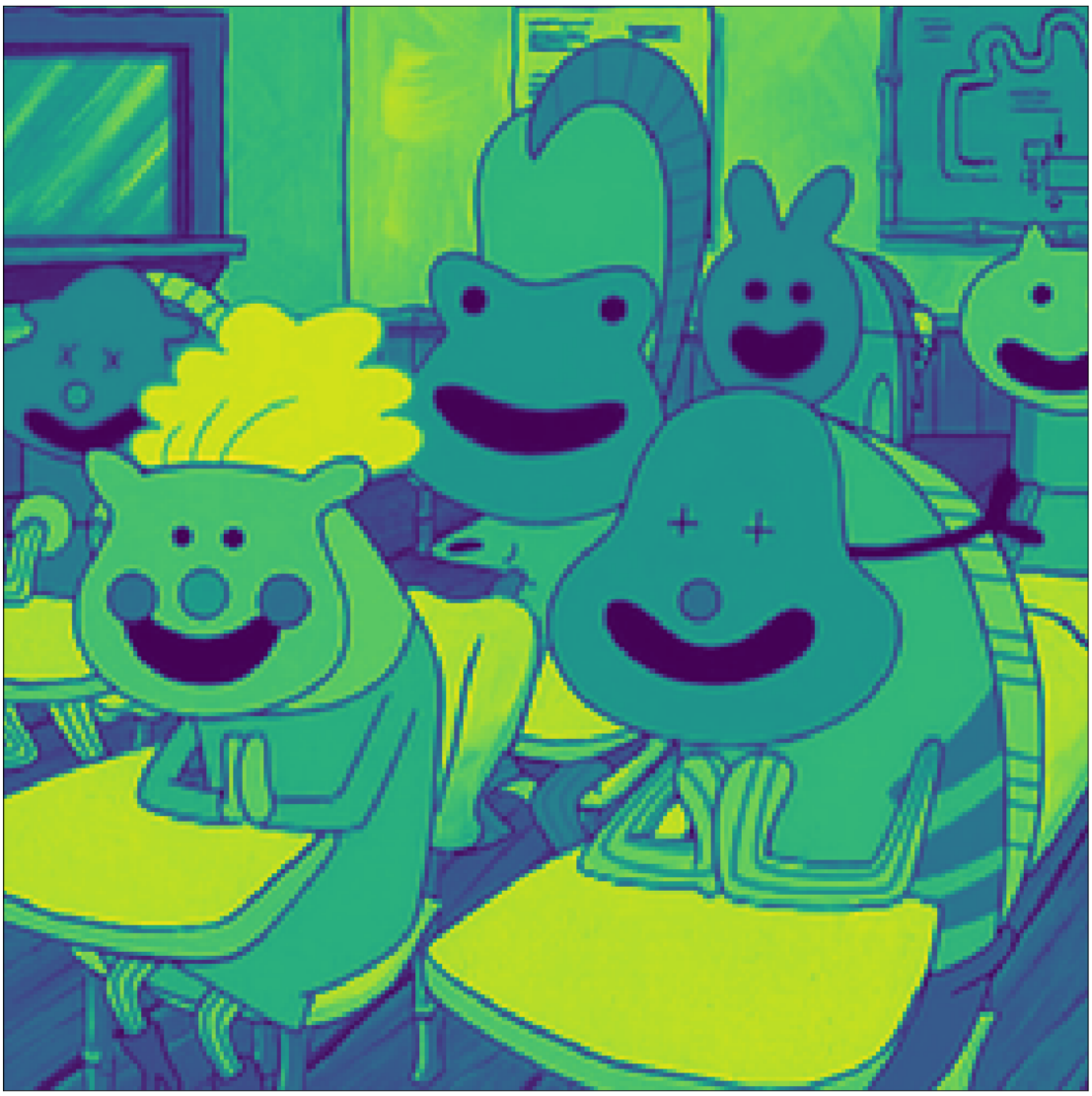
0.1.8 4. Transformasi Grayscale dan Persamaan Histogram

0.1.9 Grayscale Transformation

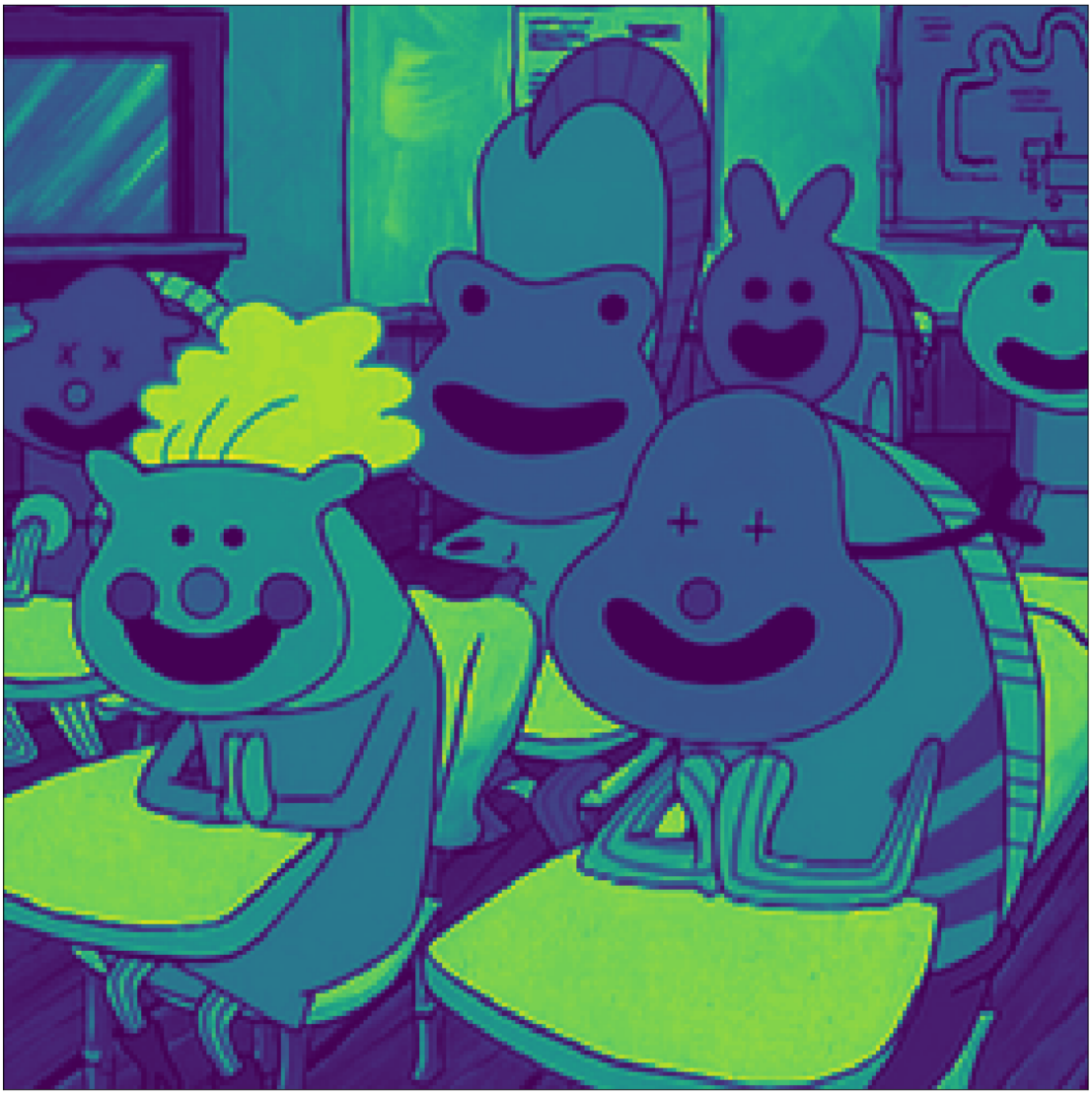
Bagian ini memberikan beberapa contoh melakukan transformasi matematis dari gambar grayscale

[21]:
```
# Ini adalah operasi kebalikan dari gambar grayscale,
# Anda bisa melihat bahwa piksel cerah menjadi gelap, dan piksel gelap menjadi␣
↪cerah
im2 = 255 - gray_image
cv_imshow(im2)
```

[22]:
```
# Transformasi gambar lainnya, setelah menambahkan konstanta,
# semua piksel menjadi lebih cerah dan efek gambar seperti perpeloncoan␣
 ↪dihasilkan
im3 = (100.0/255)*gray_image + 100
cv_imshow(im3)
```

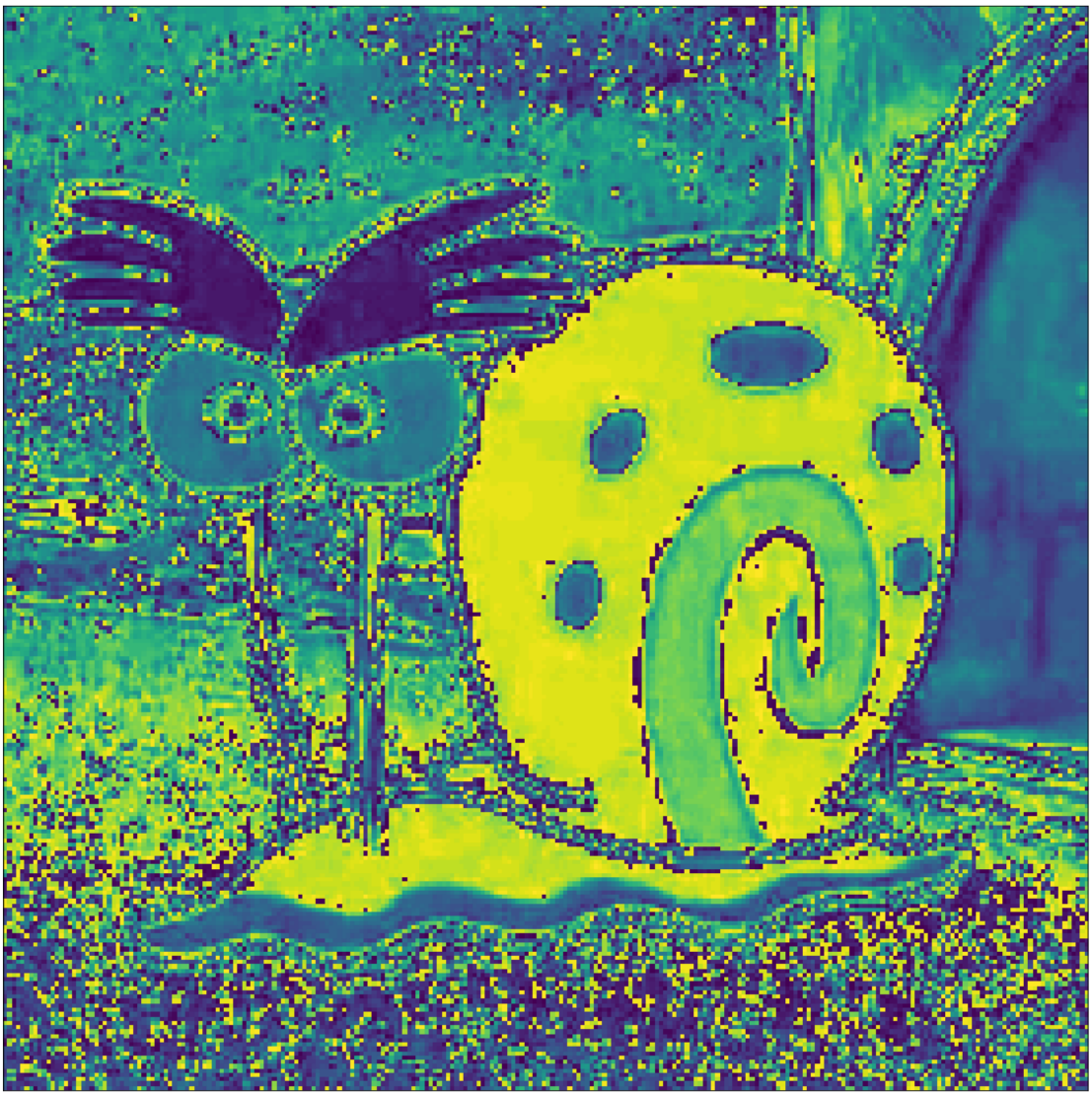


[23]:
```
# Tingkat kecerahan gambar abu-abu berkurang setelah langkah ini
im4 = 255.0*(gray_image/255.0)**2
cv_imshow(im4)
```



0.1.10 Tugas 4: Cobalah beberapa operasi matematika pada gambar Anda

[24]:
```
# Terapkan kode Anda di sini
myGrayImgTrans = myGrayImg*4
cv_imshow(myGrayImgTrans)
```

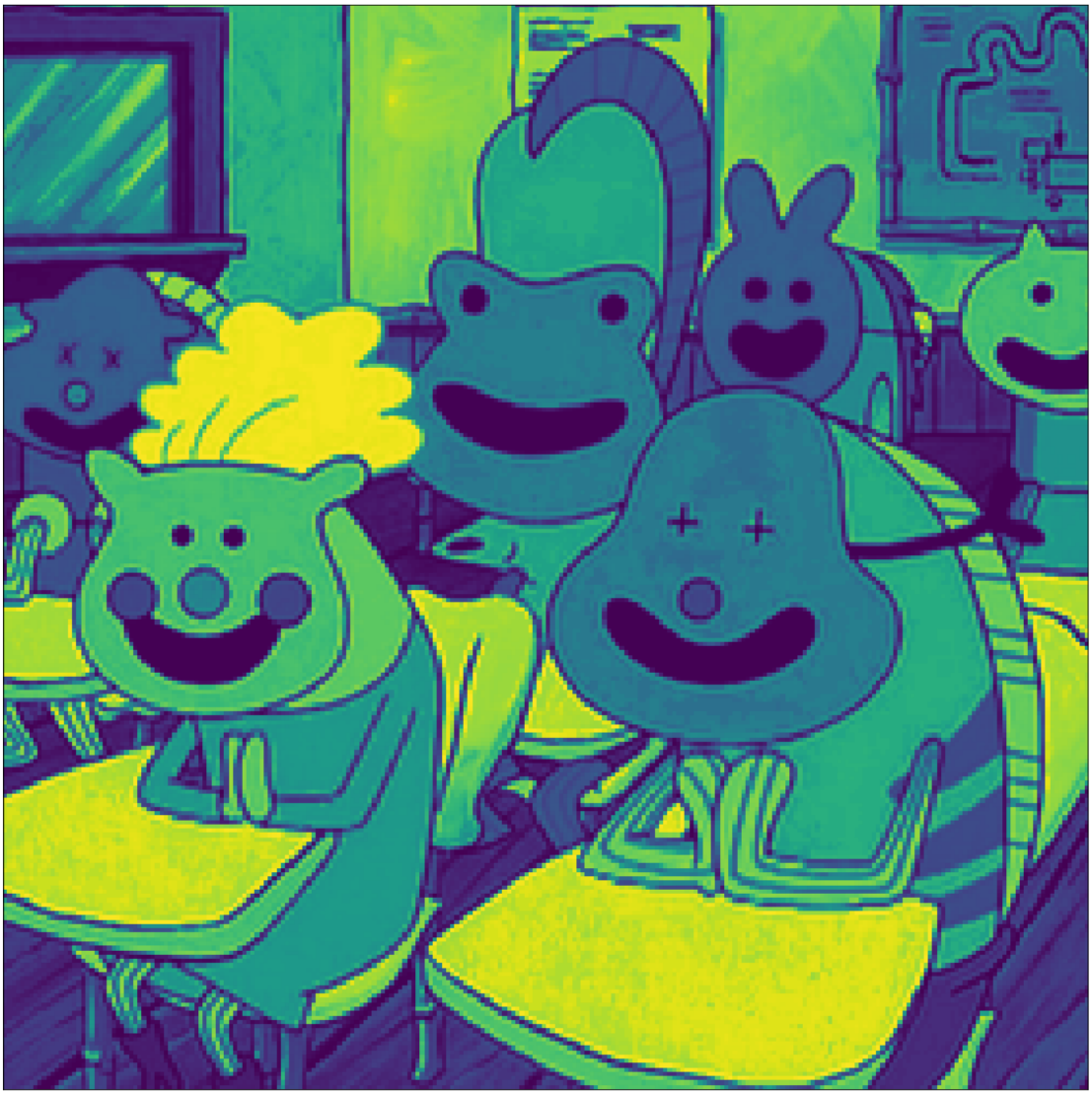


0.1.11 Histogram Equalization

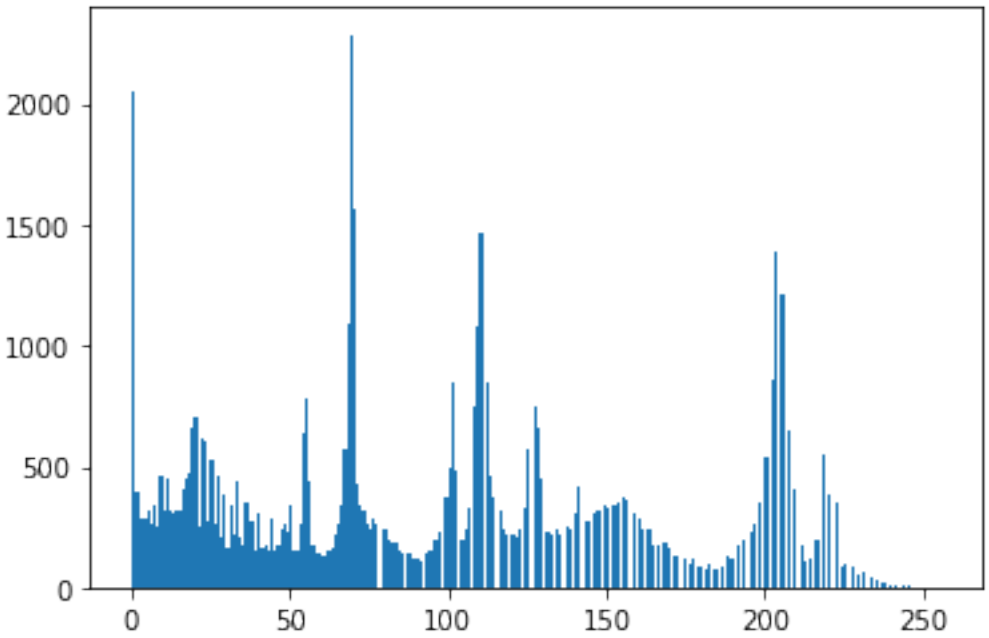
Bagian ini mendemonstrasikan pemerataan histogram pada gambar gelap. Transformasi ini meratakan histogram tingkat abu-abu sehingga semua intensitas menjadi seumum mungkin. Fungsi transformasi adalah fungsi distribusi kumulatif (cdf) dari nilai piksel pada citra (dinormalisasi untuk memetakan rentang nilai piksel ke rentang yang diinginkan). Contoh ini menggunakan gambar 4 (im4).

[25]:
```
# Fungsi dari histogram equalization
def histeq(im, nbr_bins = 256):
  """ Persamaan histogram dari citra grayscale.  """
  # dapatkan histogram gambar
```

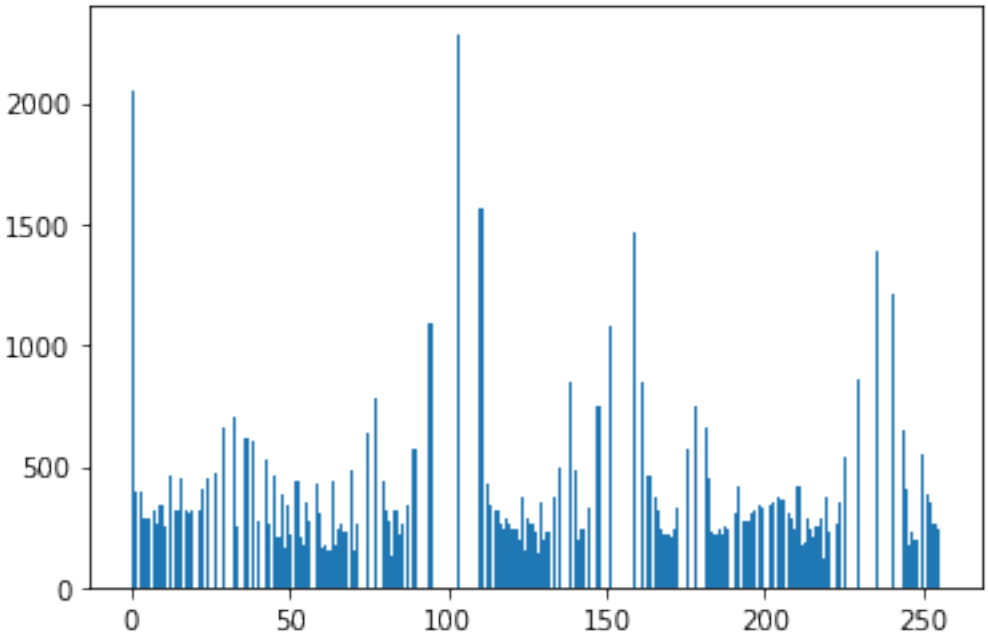
```
  imhist, bins = np.histogram(im.flatten(), nbr_bins, [0, 256])
  cdf = imhist.cumsum() # fungsi distribusi kumulatif
  cdf = imhist.max()*cdf/cdf.max()  #normalisasi
  cdf_mask = np.ma.masked_equal(cdf, 0)
  cdf_mask = (cdf_mask - cdf_mask.min())*255/(cdf_mask.max()-cdf_mask.min())
  cdf = np.ma.filled(cdf_mask,0).astype('uint8')
  return cdf[im.astype('uint8')]

# terapkan fungsi pada gambar gelap Anda untuk meningkatkan kontras
# kita dapat mengamati bahwa kontras latar belakang hitam telah meningkat
im5 = histeq(im4)
cv_imshow(im5)
```



[26]:
```
# Ekstra: mencoba memvisualisasikan histogram citra setelah pemerataan␣
↪histogram
# Sebelum pemerataan histogram
plt.hist(im4.ravel(),bins = 256, range = [0, 256])
plt.show()
```



[27]:
```
# Setelah pemerataan histogram
plt.hist(im5.ravel(),bins = 256, range = [0, 256])
plt.show()
```



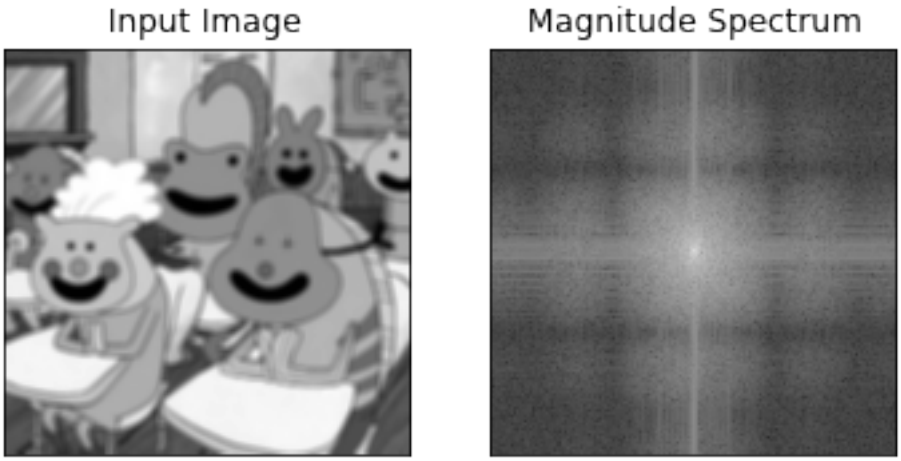
0.1.12 Tugas 5: Lakukan pemerataan histogram pada gambar grayscale Anda

[28]:
```
## TODO: Masukkan Kode Anda di sini
im6 = histeq(myGrayImg)
cv_imshow(im6)
```

0.1.13 5. Transformasi Fourier dari Gambar Abu-abu

Transformasi fourier digunakan untuk mencari domian frekuensi gambar. Anda dapat menganggap gambar sebagai sinyal yang diambil sampelnya dalam dua arah. Jadi mengambil transformasi fourier di kedua arah X dan Y memberi Anda representasi frekuensi gambar. Untuk sinyal sinusoidal, jika amplitudo berubah sangat cepat dalam waktu singkat, dapat dikatakan itu adalah sinyal frekuensi tinggi. Jika bervariasi perlahan, itu adalah sinyal frekuensi rendah. Tepi dan noise adalah konten frekuensi tinggi dalam gambar karena berubah secara drastis dalam gambar.

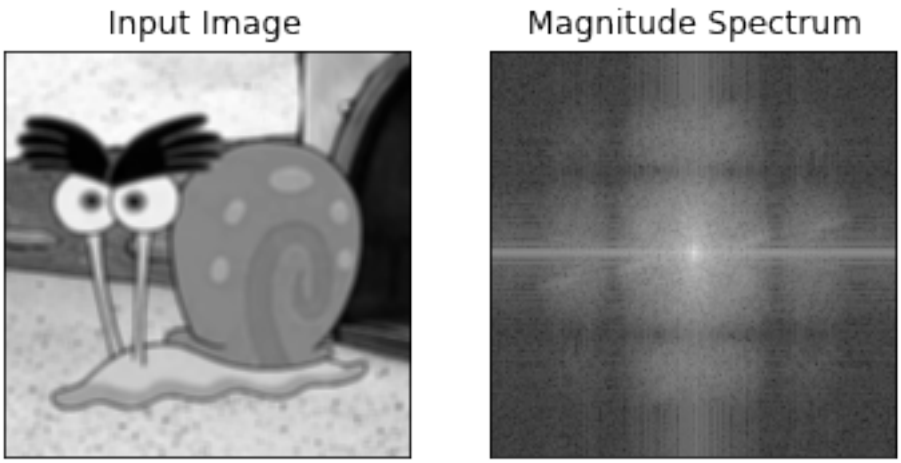
[29]:
```
# Buramkan gambar grayscale dengan filter Guassian dengan ukuran kernel 10
imBlur = cv.blur(gray_image,(5,5))
# Ubah gambar menjadi domain frekuensi
f = np.fft.fft2(imBlur)
# Bawa komponen frekuensi-nol ke tengah
fshift = np.fft.fftshift(f)
magnitude_spectrum = 30*np.log(np.abs(fshift))

plt.subplot(121),plt.imshow(imBlur, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```



0.1.14 Tugas 6: Hasilkan transformasi fourier dari gambar grayscale Anda

[30]:
```
# todo: Terapkan kode Anda di sini
# Buramkan gambar grayscale dengan filter Guassian dengan ukuran kernel 10
imBlur = cv.blur(myGrayImg,(5,5))
# Ubah gambar menjadi domain frekuensi
f = np.fft.fft2(imBlur)
# Bawa komponen frekuensi-nol ke tengah
fshift = np.fft.fftshift(f)
magnitude_spectrum = 30*np.log(np.abs(fshift))

plt.subplot(121),plt.imshow(imBlur, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(magnitude_spectrum, cmap = 'gray')
```

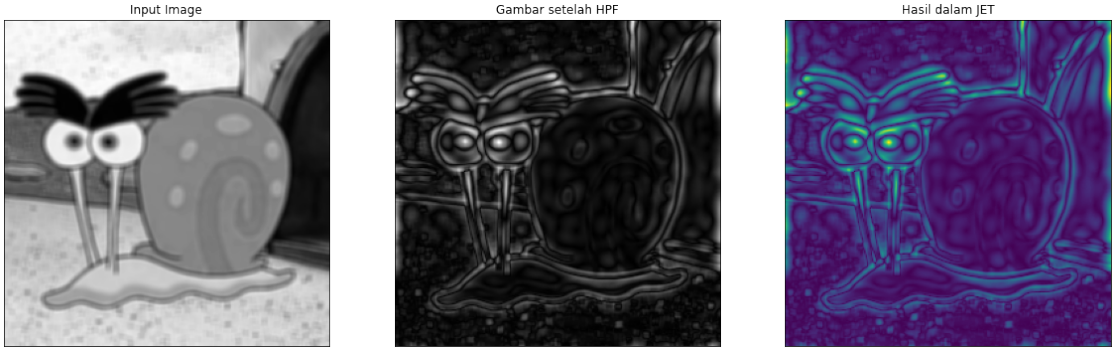
```
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```



0.1.15 6. Menemukan Tepi dengan Highpass Filtering di FFT

Bagian ini mendemonstrasikan melakukan high pass filter untuk menghilangkan komponen frekuensi rendah, sehingga menghasilkan gambar yang tajam yang berisi tepinya.

[31]:
```
rows, cols = imBlur.shape
crow,ccol = round(rows/2) , round(cols/2)
# hilangkan frekuensi rendah dengan ukuran persegi panjang 10
fshift[crow-10:crow+10, ccol-10:ccol+10] = 0
f_ishift = np.fft.ifftshift(fshift)
img_back = np.fft.ifft2(f_ishift)
img_back = np.abs(img_back)

plt.figure(figsize=([20, 20]))
plt.subplot(131),plt.imshow(imBlur, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(132),plt.imshow(img_back, cmap = 'gray')
plt.title('Gambar setelah HPF'), plt.xticks([]), plt.yticks([])
plt.subplot(133),plt.imshow(img_back)
plt.title('Hasil dalam JET'), plt.xticks([]), plt.yticks([])
plt.show()
```


Input Image
Gambar setelah HPF
Hasil dalam JET