Nama : Fadia Ramadhana
NIM : G64170026

LKP 8
PENGOLAHAN CITRA DIGITAL

1. Download image daun yang berpenyakit dibawah ini:

2. Tampilkan spektrum fourier untuk mendeteksi bercak penyakit pada daun tersebut dan jelaskan berdasarkan spektrum yang kamu hasilkan!

Fast Fourier Transform (FFT) diterapkan dalam berbagai bidang dari pengolahan sinyal digital dan memecahkan persamaan diferensial parsial menjadi algoritma-algoritma untuk penggandaan bilangan integer dalam jumlah banyak. Ada pun kelas dasar dari algoritma ***FFT*** yaitu decimation in time (DIT) dan decimation in frequency (DIF). Garis besar dari kata Fast diartikan karena formulasi ***FFT*** jauh lebih cepat dibandingkan dengan metode perhitungan algoritma Fourier Transform sebelumnya. Karena ***Discrete Fourier Transform*** memisahkan inputnya menjadi komponen yang berkontribusi pada frekuensi diskrit, ia memiliki sejumlah besar aplikasi dalam pemrosesan sinyal digital, misalnya untuk penyaringan, dan dalam konteks ini input yang diskrit ke transformasi biasanya disebut sebagai sinyal , yang ada dalam domain waktu. Outputnya disebut spektrum atau transformasi dan ada di domain frekuensi.

$$A_k = \sum_{m=0}^{n-1} a_m \exp\left\{-2\pi i \frac{mk}{n}\right\} \qquad k = 0, \ldots, n-1.$$

The inverse DFT is defined as

$$a_m = \frac{1}{n} \sum_{k=0}^{n} {}^{1} A_k \exp\left\{2\pi i \frac{mk}{n}\right\} \qquad m = 0, \ldots, n-1.$$

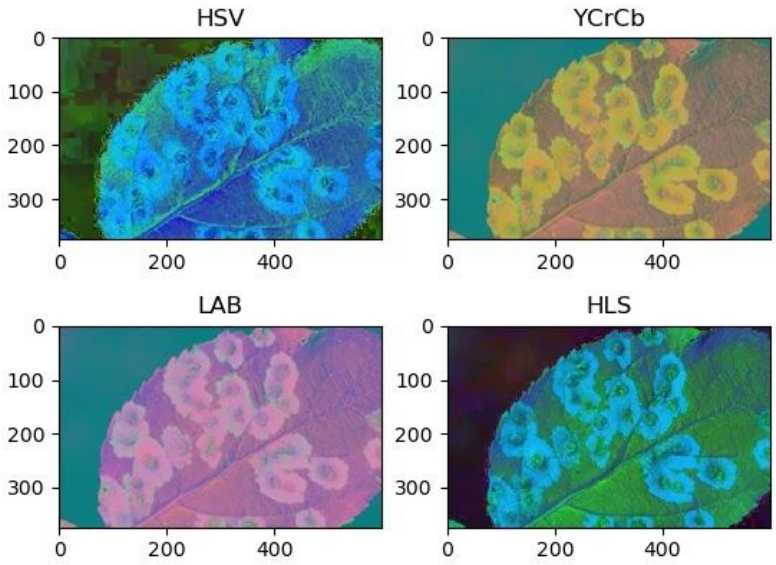
Image Preprocessing :

1. **Melakukan *crop* image dan *smoothing* dengan Gaussian Blur**

Saya melakukan crop pada image karena pada image asli di bagian sudut kanan bawah terdapat objek yang warnanya hampir mirip dengan warna penyakit pada daun namun saya tidak dapat memastikan apakah itu adalah penyakit atau bukan. Gaussian blur digunakan untuk melakukan smoothing dari image sehingga meminimalisir noise ketika akan diproses nantinya.

2. **Mengubah color space image menjadi berbagai jenis color space (HSV, YCrCb, LAB, dan HSL)**

```
# konversi image dari RGB menjadi beberapa colorspace
imageHSV = cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
imageYCrCb = cv2.cvtColor(img,cv2.COLOR_BGR2YCR_CB)
imageLAB = cv2.cvtColor(img,cv2.COLOR_BGR2LAB)
imageHLS = cv2.cvtColor(img,cv2.COLOR_BGR2HLS)
```



Dari hasil output yang didapat dari mengubah colorspace, saya akan memilih color space YCrCb karena hasil yang dihasilkan cukup dapat memberikan perbedaan yang signifikan antara bagian daun yang sehat dan bagian daun yang terdapat penyakit. Selain itu hasil dari color space YCrCb memiliki image yang lebih smooth dibandingkan dengan HSV dan HLS yang memiliki banyak noise.

3. Melakukan segmentasi penyakit pada daun

Untuk melakukan segmentasi, hal yang harus dilakukan adalah mengeset range nilai piksel sebagai threshold untuk masing-masing channel yaitu Y, Cr, dan Cb. Lower (min) range merupakan sebuah array yang menyimpan batas bawah dari masing-masing channel. Upper (max) range merupakan sebuah array yang menyimpan batas atas dari masing-masing channel. OpenCV menggunakan range YCrCb sebagaimana value aslinya yaitu :

- 0 > Y > 255
- 0 > Cr > 255
- 0 > Cb > 255

Bersamaan dengan aturan tersebut dan menggunakan paper dan beberapa website rujukan dengan beberapa tes empiris dan evaluasi visual, saya akan melakukan percobaan dengan menggunakan range untuk Y, Cr, dan Cb yaitu:
(0 - 255, 135 - 255, 0 - 255)

```
29  # batas bawah YCrCb
30  min_ycrcb = np.array([0,135,0],np.uint8)
31  # batas atas YCrCb
32  max_ycrcb = np.array([255,255,255],np.uint8)
```

Gambar yang telah diubah color space nya menjadi YCrCb, akan digunakan untuk membuat mask bersama dengan nilai range (min dan max) dari

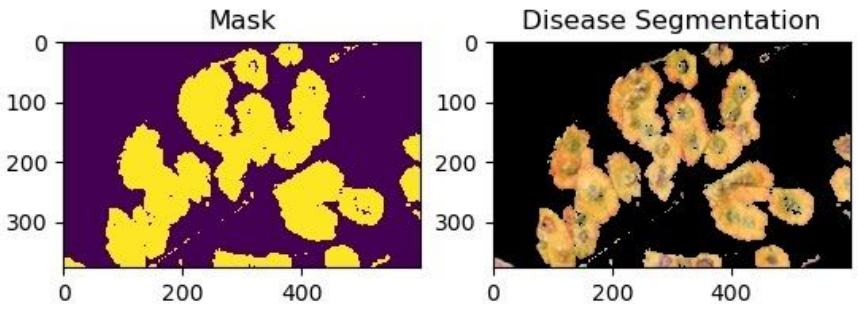
masing-masing channel menggunakan fungsi cv2.inRange. Mask untuk image didapat menggunakan fungsi dari openCV yaitu fungsi inRange dimana fungsi tersebut hanya mengembalikan MASK bernilai biner. Piksel putih (255) mewakili piksel yang termasuk ke dalam range antara min dan max dan begitu pula piksel hitam (0) mewakili piksel yang tidak termasuk ke dalam range. Kemudian langkah selanjutnya adalah melakukan segmentasi wajah pada gambar asli dengan menggunakan mask yang telah didapatkan pada step sebelumnya dengan memanfaatkan fungsi **cv2.bitwise_and()** yang menjaga setiap piksel dalam gambar yang diberikan jika nilai yang sesuai dalam mask adalah 1.

```
# membuat mask dengan
# mengembalikan nilai biner image menggunakan range bawah dan atas (min, max)
Mask = cv2.inRange(imageYCrCb, min_ycrcb, max_ycrcb)

# menggabungkan gambar asli dengan mask
# mask berada di atas gambar asli
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
segmentation = cv2.bitwise_and(img, img, mask = Mask)
```

.

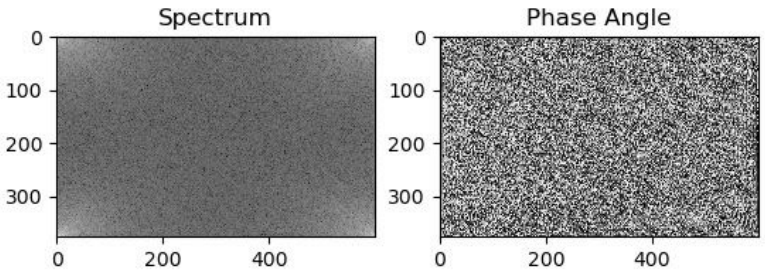
Hasil dari segmentasi penyakit pada daun tersebut adalah sebagai berikut :



Hasil segmentasi yang telah diperoleh tersebut akan diubah colorspace nya menjadi grayscale kemudian menjadi input untuk dicari spektrum nya menggunakan algoritma *Fast Fourier Transform* (FFT).

Algoritme Fast Fourier Transform :

1. **Menghitung Fast Fourier Transform (FFT) 2-dimensi**
   Hasil dari proses FFT adalah array bilangan kompleks yang sangat sulit untuk divisualisasikan secara langsung. Oleh karena itu, kita harus mengubahnya menjadi ruang 2 dimensi. Berikut adalah dua cara yang dapat kita memvisualisasikan hasil FFT ini: 1. Spectrum 2.Phase angle .



Dari gambar spektrum, ada beberapa pola simetris di empat sudut. Pola-pola ini dapat ditranslasikan ke tengah gambar di langkah berikutnya. Area putih pada gambar spektrum menunjukkan daya frekuensi yang tinggi. Sudut-sudut pada gambar spektrum mewakili frekuensi rendah. Di sisi lain, sulit untuk mengidentifikasi pola yang terlihat dari gambar *phase angle*. Ini tidak menunjukkan bahwa sudut fase FFT sama sekali tidak berguna karena fase mempertahankan karakteristik bentuk yang merupakan informasi yang sangat diperlukan untuk gambar.

Code :

```
segmentation = cv2.cvtColor(segmentation, cv2.COLOR_BGR2GRAY)

original = np.fft.fft2(segmentation)
plt.subplot(121), plt.imshow(np.log(np.abs(original)), "gray"), plt.title("Spectrum")

plt.subplot(122), plt.imshow(np.angle(original), "gray"), plt.title("Phase Angle")
plt.show()
```

np.fft.fft2() merupakan fungsi yang dapat digunakan untuk menghitung Discrete Fourier Transform (DFT) dua dimensi. Kemudian akan digunakan logaritma nilai absolut pada hasil DFT.

2. **Melakukan *shift* pada komponen dengan *zero-frequency* ke bagian tengah spektrum**

   2D FFT memiliki properti translasi dan rotasi, sehingga kita dapat menggeser frekuensi tanpa kehilangan informasi apa pun. Dengan menggeser komponen *zero-frequency* ke pusat spektrum, gambar spektrum akan lebih terlihat bagi manusia. Selain itu, dengan melakukan translasi dapat membantu mempermudah jika ingin menerapkan *high pass filter* dan *low pass filter*. Jangan lupa untuk menambahkan 1 ke hasil spektrum yang telah diberi nilai absolute, karena argumen ke fungsi logaritma harus lebih besar dari atau sama dengan 1.0. Juga perhatikan bahwa tipe data adalah 8-bit unsigned-int, karena kita ingin menampilkannya sebagai gambar. Kita dapat menggunakan fungsi **np.fft.fftshift()**

```
center = np.fft.fftshift(original)
plt.subplot(153), plt.imshow(np.log(1+np.abs(center)), cmap="gray")
plt.title("Centered Spectrum")
```

3. **Melakukan *inverse* pada output yang dihasilkan step 2. *Shift* kembali komponen dengan *zero-frequency* ke lokasi asalnya**

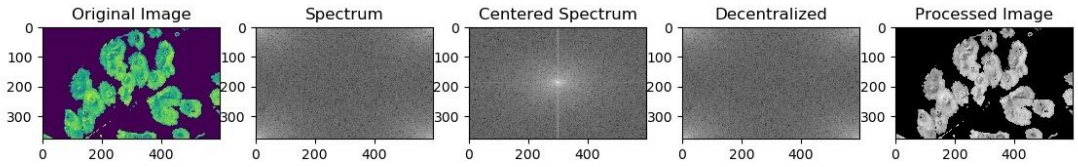
```
inv_center = np.fft.ifftshift(center)
plt.subplot(154), plt.imshow(np.log(1+np.abs(inv_center)), cmap="gray"), plt.title("Decentralized")
```

Proses langkah 3 mengubah informasi dari spektrum kembali ke gambar dengan colorspace grayscale menggunakan fungsi **np.fft.ifftshift()**

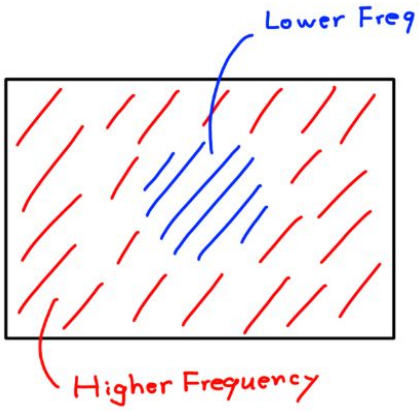
4. **Melakukan *inverse* pada output yang dihasilkan step 1. Menghitung *inverse* dari FFT 2-dimensi**
   Inverse Fourier Transform (IFT) mengambil gambar domain frekuensi sebagai input dan kemudian mengembalikan gambar asli. Kita dapat menggunakan transformasi terbalik ini untuk menerapkan beberapa teknik menarik pada gambar, seperti Low Pass Filter (LPF) atau estimasi gerakan kamera. Kita dapat menggunakan fungsi **np.fft.ifft2()** yang akan melakukan Inverse Fourier Transform dua dimensi

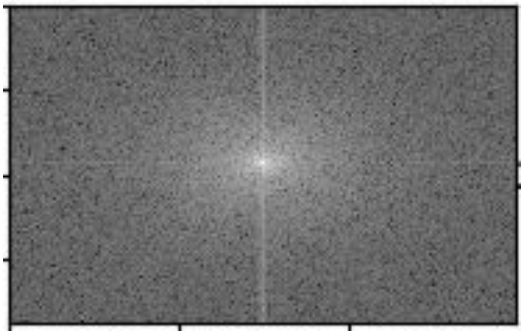
```
processed_img = np.fft.ifft2(inv_center)
plt.subplot(155), plt.imshow(np.abs(processed_img), cmap="gray")
plt.title("Processed Image")
```

Output Final :



Interpretasi :





Dapat melihat bahwa gambar hasil dari contoh di atas memiliki piksel paling terang (nilai lebih besar) di tengah dan piksel ukuran luar relatif kurang cerah (nilai lebih kecil). Output dari domain frekuensi image memberikan informasi seberapa banyak setiap komponen frekuensi yang ada dalam image asli nya. Piksel yang berada di dekat pusat (center) merepresentasikan komponen yang berfrekuensi lebih rendah, dan piksel pada bagian terluar merepresentasikan komponen yang berfrekuensi lebih tinggi. Dengan demikian jika piksel terletak di dekat pusat (center) lebih terang dibandingkan dengan yang berada di bagian terluar, ini artinya image asli sebagian besar disusun oleh komponen yang memiliki frekuensi lebih rendah daripada komponen yang memiliki frekuensi lebih tinggi.
Penyakit daun yang telah disegmentasi dari image leaf-spot-disease-control-in-michigan.jpg, komponen yang memeiliki frekuensi lebih rendah muncul sebagai objek besar yang bertekstur kasar. Secara umum, komponen frekuensi yang lebih tinggi dapat menambahkan beberapa detail ke dalam gambar. Objek kecil dan edge merupakan objek yang sebagian besar disusun oleh komponen dengan frekuensi yang lebih tinggi.

Sumber Referensi :

https://medium.com/@hicraigchen/digital-image-processing-using-fourier-transform-in-python-bcb49424fd82

https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html

https://docs.scipy.org/doc/numpy/reference/routines.fft.html