

LAPORAN UAS



Reviewer : **Kelompok 4**

Nama Anggota : Melania Syafrida P. (18081010011),
Dwi Wahyu Effendi (18081010017),
Dedy Ramadhan (18081010020),
Amirah Aulia Fitri (18081010029),
Quincy Mayferta P.P.A (19081010127)

Link Video Presentasi dan Demo Program :

[UAS PCD Identifikasi Daun Tanaman Buah Tomat](#)

Judul : Identifikasi Citra Daun Tanaman Tomat Menggunakan Ekstraksi Fitur GLCM, Klasifikasi KNN dan Menghitung Tingkat Kerusakan Daun menggunakan Metode Laplacian Of Gaussian

Latar Belakang

Permasalahan pada daun tanaman tomat tidak hanya muncul dari bentuk buah dan kualitas tomat tersebut, tapi permasalahan juga muncul dari beberapa penyakit atau hama yang menyerang dari beberapa bagian tanaman tomat itu sendiri seperti pada bagian daun tomat yang akan mempengaruhi pertumbuhan tanaman tomat tersebut.

Ekstraksi fitur GLCM dan Klasifikasi KNN digunakan untuk mengklasifikasi daun tanaman tomat tersebut sehat atau sakit. lalu kita mendeteksi tepi menggunakan metode Laplacian of Gaussian (LoG). Setelah mendeteksi tepi, didapat nilai rata-rata Mean Square Error dari daun tomat sehat dan daun tomat sakit

Metode :

Tahapan metode yang digunakan di program ini dimulai dari design program, lalu kita melakukan ekstraksi fitur dengan menggunakan GLCM dan Klasifikasi KNN untuk mendeteksi daun sehat dan daun sakit, selanjutnya kita mendeteksi tepi menggunakan Log (Laplacian of Gaussian) untuk menghitung MSE (Mean Square Error).

1. Design Program

Pada Program untuk mengetahui daun tomat tersebut sehat atau rusak, kita melakukan Pengolahan citra dengan tahapan ekstraksi fitur tekstur menggunakan algoritma GLCM dan untuk mengklasifikasi citra tersebut sehat atau rusak kita menggunakan Klasifikasi KNN. Setelah itu untuk menghitung MSE (Mean Square Error) dari citra daun tomat. sehat atau rusak kita harus mengubah citra berwarna ke grayscale. lalu selanjutnya kita menggunakan Gaussian Filter untuk memperhalus atau (blur) citra daun tomat. lalu selanjutnya kita menggunakan deteksi tepi laplacian untuk mendeteksi tepi daun tomat. lalu baru kita bisa menghitung nilai MSE dari citra daun tomat tersebut

2. GLCM

Gray Level Co-occurrence Matrix adalah suatu matrik korelasi yang elemennya merupakan jumlah kemunculan piksel-piksel yang memiliki nilai tingkat keabuan tertentu, di mana pasangan piksel itu berada pada jarak (d) dan sudut tertentu (Θ). Orientasi sudut yang digunakan adalah yaitu sudut $0^\circ, 45^\circ, 90^\circ, 135^\circ, 180^\circ$ dst Sedangkan jarak antar piksel biasanya ditetapkan sebesar 1 piksel, atau 2 piksel

2.1 Energy

Energi menyatakan ukuran konsentrasi pasangan dengan intensitas keabuan tertentu pada matriks. Nilai energi (E) dapat dihitung dengan persamaan berikut

$$E = \sum_{i,j} (p_{i,j})^2 \quad (1)$$

Dengan :

i = tingkat keabuan baris ke – i

j = tingkat keabuan kolom ke – j

$p_{i,j}$ = Peluang keabuan baris ke – i , kolom ke – j

2.2. Kontras

Kontras menyatakan perbedaan intensitas antara nilai tertinggi (terang) dan nilai terendah (gelap) dari sepasang piksel yang saling berdekatan.

$$CON = \sum_{i,j} i - j^2 p_{i,j} \quad (3)$$

Dengan :

i = tingkat keabuan baris ke – i

j = tingkat keabuan kolom ke – j

$p_{i,j}$ = Peluang keabuan baris ke – i , kolom ke – j

2.3 Homogenitas

Nilai homogenitasnya ditentukan dari derajat keabuan yang sejenis.

$$H = \sum_{i,j} \frac{p_{i,j}}{1+|i-j|}$$

2.4 Correlation

Correlation digunakan untuk mengukur kemungkinan terjadinya gabungan dari pasangan piksel yang ditentukan.

$$\sum_{i,j=0}^{levels-1} P_{i,j} \left[\frac{(i - \mu_i)(j - \mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}} \right]$$

3. Gaussian Filter

Filter Gaussian sangat baik untuk menghilangkan noise yang bersifat sebaran normal, filter Gaussian menempatkan warna transisi yang signifikan dalam sebuah image, kemudian membuat warna-warna pertengahan untuk menciptakan efek lembut pada sisi-sisi sebuah image. Berikut rumus Gaussian Filter :

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

4. Grayscale

Grayscale adalah warna-warna piksel yang berada dalam rentang gradasi warna hitam dan putih. Mengubah warna ke grayscale pada pengubahan sebuah gambar menjadi grayscale dapat dilakukan dengan cara mengambil semua pixel pada gambar kemudian warna tiap pixel akan diambil informasi mengenai 3 warna dasar yaitu merah, biru dan hijau (melalui fungsi warnatoRGB), ketiga warna dasar ini akan dijumlahkan. Berikut contoh perhitungan grayscale :

$$\begin{aligned}
 Grayscale &= (0,2989 \times R) + (0,5870 \times G) \\
 &\quad + (0,1140 \times B) \\
 &= (0,2989 \times 217) + (0,5870 \times 224) \\
 &\quad + (0,1140 \times 219) \\
 &= 62,8866 + 131,448 + 24,966 \\
 &= 219,3006
 \end{aligned}$$

Perhitungan yang sama dilakukan pada baris dan kolom lainnya.

5. Laplacian

Operator turunan kedua disebut juga operator Laplace. Operator Laplace mendeteksi lokasi tepi lebih akurat khususnya pada tepi yang curam. Pada tepi yang curam, turunan keduanya mempunyai persilangan nol (zero-crossing), yaitu titik di mana terdapat pergantian tanda nilai turunan kedua dari positif ke negative atau sebaliknya. Berikut adalah rumus perhitungan laplacian :

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

6. MSE (Mean Square Error)

Mean Squared Error (MSE) adalah Rata-rata Kesalahan kuadrat diantara nilai aktual dan nilai peramalan. Metode Mean Squared Error secara umum digunakan untuk mengecek estimasi berapa nilai kesalahan pada peramalan. Nilai Mean Squared Error yang rendah atau nilai mean squared error mendekati nol menunjukkan bahwa hasil peramalan sesuai dengan data aktual dan bisa dijadikan untuk perhitungan peramalan di periode mendatang. Metode Mean Squared Error biasanya digunakan untuk mengevaluasi metode pengukuran dengan model regresi atau model peramalan seperti Moving Average, Weighted Moving Average dan Analisis Trendline.

Cara menghitung Mean Squared Error (MSE) adalah melakukan pengurangan nilai data aktual dengan data peramalan dan hasilnya dikuadratkan (squared) kemudian dijumlahkan secara keseluruhan dan membaginya dengan banyaknya data yang ada. Berikut rumus perhitungan MSE :

$$MSE = \frac{\sum_{t=1}^n (At - Ft)^2}{n}$$

7. K-Nearest Neighbor (KNN)

K-nearest neighbors atau knn adalah algoritma yang berfungsi untuk melakukan klasifikasi suatu data berdasarkan data pembelajaran (train data sets), yang diambil dari k tetangga terdekatnya (nearest neighbors). Dengan k merupakan banyaknya tetangga terdekat.

Cara kerja K-Nearest Neighbor (KNN) secara umum adalah menentukan jumlah tetangga (K) yang akan digunakan untuk pertimbangan penentuan kelas, hitung jarak dari data baru ke masing-masing data point di dataset, ambil sejumlah K data dengan jarak terdekat, kemudian tentukan kelas dari data baru tersebut.

Proses klasifikasi kelas K-Nearest Neighbor (KNN) dilakukan dengan mencari titik terdekat dari data baru (nearest neighbor). Tujuannya adalah untuk mendapatkan data sebanyak k dengan jarak terdekat dengan data baru. Teknik pencarian tetangga terdekat yang umum dilakukan dengan menggunakan formula jarak euclidean. Berikut adalah rumus jarak euclidean:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2}$$

Code :

```
%Data Train  
cd('D:\MELA\citra\tomato\train');  
dataset={'tomat rusak', 'tomat sehat'};  
jumlah_kelas=length(dataset);
```

- Kode diatas digunakan untuk menentukan direktori folder yang membaca data train yang sudah kita pisahkan berdasarkan jenis kelas dan menghitung banyaknya jumlah data yang akan kita train.

```
%GLCM  
for i=1:jumlah_data  
    file=citra(i).name;  
    convertcitra=rgb2gray(imread(file));  
  
    fitur = graycoprops(graycomatrix(convertcitra));  
    fiturmatriks(i+jumlah_data*(n-1),1)=fitur.Contrast;  
    fiturmatriks(i+jumlah_data*(n-1),2)=fitur.Correlation;  
    fiturmatriks(i+jumlah_data*(n-1),3)=fitur.Energy;  
    fiturmatriks(i+jumlah_data*(n-1),4)=fitur.Homogeneity;  
  
    kelas(i+jumlah_data*(n-1))=n;  
end  
cd('..');  
end
```

- Kode diatas digunakan untuk melakukan convert citra rgb ke grayscale. Setelah citra selesai di convert akan dilakukan ekstraksi fitur yaitu ekstraksi fitur contrast, fitur correlation, fitur energy dan fitur homogeneity pada citra data train satu persatu.

```
%Testing%  
model=fitcknn(fiturmatriks,kelas');  
cd('testing')  
for j=1:2  
    nama=sprintf('D%d.jpg',j); %Data Testing%  
    a=rgb2gray(imread(nama));  
    m=graycomatrix(a);  
    g=graycoprops(m);  
    %imshow(a);  
    uji(j,1)=g.Contrast;  
    uji(j,2)=g.Correlation;  
    uji(j,3)=g.Energy;  
    uji(j,4)=g.Homogeneity;  
  
    target(j)=2;  
    klasifikasi(j)=model.predict(uji(j,:)); %prediksi  
    if klasifikasi(j)==target(j)  
        hasil(j)={'Sehat'};  
    else  
        hasil(j)={'Rusak'};  
    end
```

end

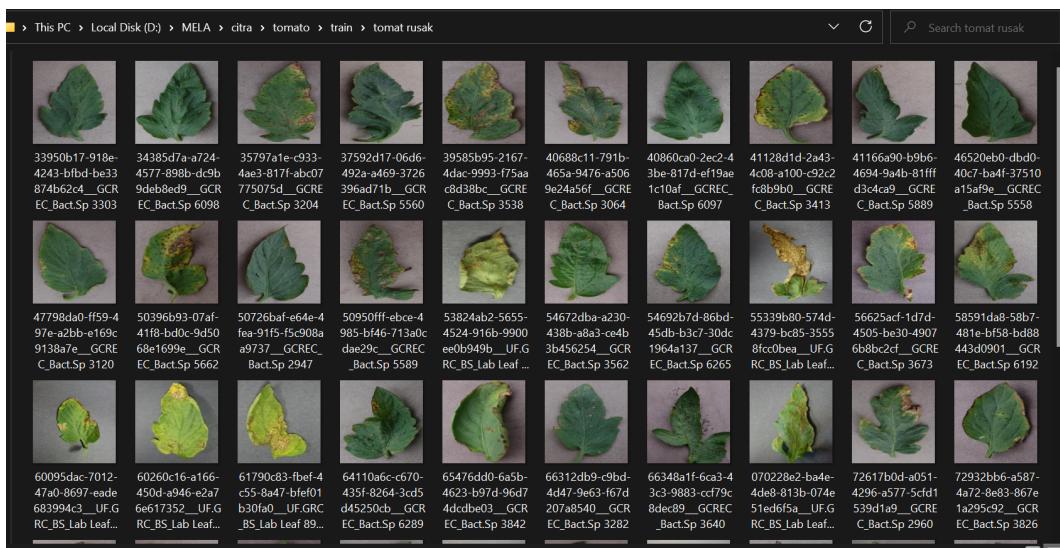
```
[{'Contrast','Correlation','Energy','Homogeneity','Target','Kelas','Hasil'};  
num2cell([uji target' klasifikasi']) hasil']
```

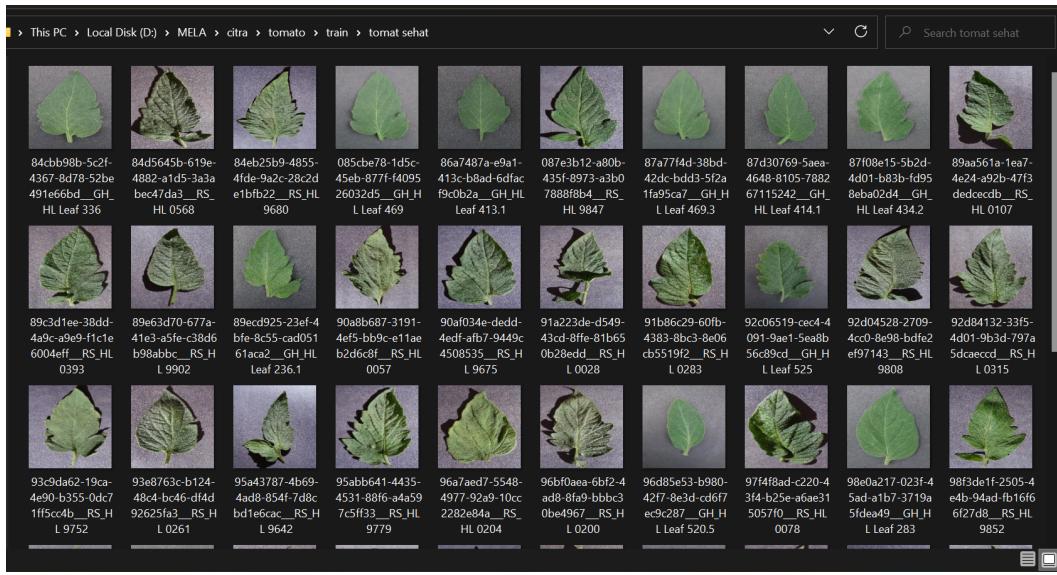
- Kode diatas digunakan untuk melakukan testing pada citra untuk melakukan prediksi apakah citra yang di testing dapat diprediksi dengan benar. Sebelum itu, data testing akan masuk ke direktori data testing dimana file testing berbeda dengan file data train. Citra RGB pada data testing akan dilakukan convert citra terlebih dahulu dan dilakukan ekstraksi fitur.

```
for j=1:2  
    nama=sprintf('D%d.jpg',j);  
    gray=rgb2gray(imread(nama));  
  
    %konversi citra with LoG Filter  
    h=fspecial('log');  
    filtered=imfilter(gray,h);  
    %imshow(filtered);  
    %figure;  
  
    err = immse(gray(j), filtered(j)); %Mse%  
    fprintf('Mse D%d.jpg : %d\n',j, err);  
end
```

- Kode diatas digunakan untuk melakukan convert citra RGB menjadi grayscale dan mengkonversi menggunakan LoG filter atau laplacian of gaussian filter yang digunakan untuk deteksi tepi citra. Selanjutnya, akan dicari nilai MSE atau Mean Square Error pada citra data testing.

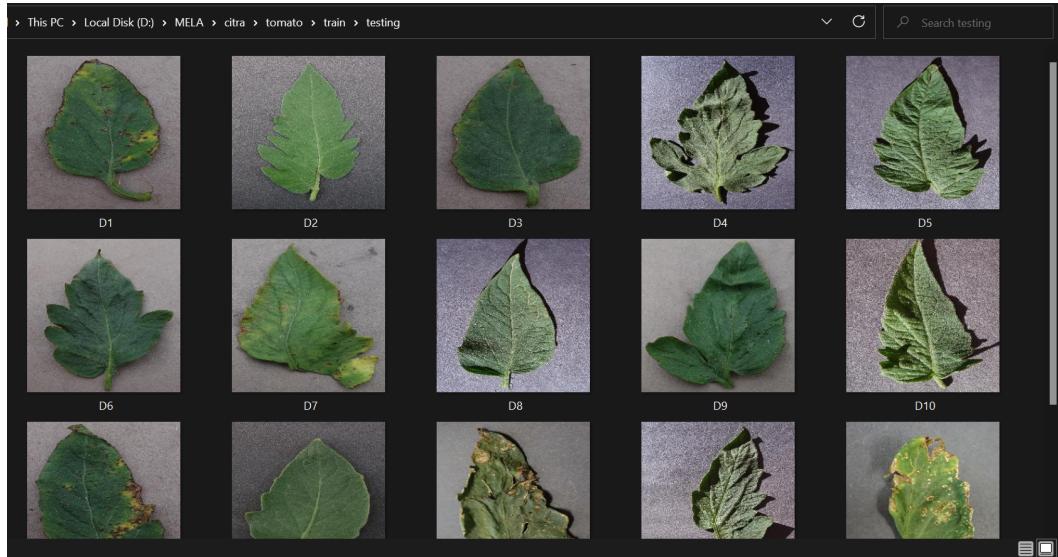
Data Train :



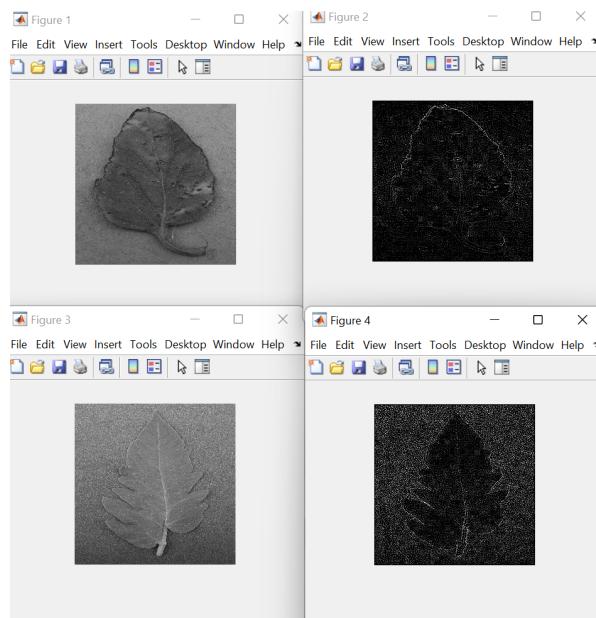


(Gambar 1.1 Data Train Daun Tomat Sehat)

Data Testing :



(Gambar 1.7 Data Testing)



(Gambar 1.2 Hasil Konversi Citra RGB)

```
{'Contrast'}  {'Correlation'}  {'Energy'}    {"Homogeneity"}  {'Target'}   {'Kelas'}    {'Hasil'}
{[ 0.2481]}  {[ 0.8563]}  {[0.1898]}  {[ 0.8853]}  {[ 2]}     {[ 1]}     {'Rusak'}
{[ 0.7398]}  {[ 0.5532]}  {[0.1300]}  {[ 0.7434]}  {[ 2]}     {[ 2]}     {'Sehat'}

Mse D1.jpg : 23104
Mse D2.jpg : 18496
```

(**Gambar 1.3** Hasil Ekstraksi Fitur GLCM dan MSE)

Kesimpulan :

1. Dari pembahasan kita dapat mengetahui citra daun mana yang sehat dan citra daun mana yang rusak melalui proses ekstraksi fitur GLCM dan Klasifikasi KNN.
2. Dari pembahasan sebelumnya juga dapat disimpulkan bahwa deteksi tepi daun tanaman tomat pada citra dapat dilakukan dengan metode Laplacian of Gaussian (LoG) dan operator Laplacian.
3. Dari proses deteksi tepi terhadap daun tomat didapat nilai rata-rata yaitu Mean Square Error yang digunakan untuk mengetahui tingkat kerusakan pada daun tomat
4. Nilai MSE akan semakin besar berdasarkan tingkat kerusakan daun.(LoG) dan operator Laplacian.