PENGOLAHAN CITRA DIGITAL IMPLEMENTASI

TEKSTUR KAYU KELAPA DENGAN PYTHON3

Kayu kelapa memiliki banyak manfaat dan dapat digunakan sebagai produk bernilai tinggi. Pada pembahasan di buku ini bertujuan untuk mengidentifikasi tekstur kayu kelapa dengan kepadatan tinggi dan sedang. Metode yang digunakan adalah dengan menggunakan segmentasi gambar ambang batas bertingkat yang kemudian di lanjutkan dengan menggunakan metode Gray Level Co-Occurrence Matrix (GLCM). Dalam hal ini, kualitas kayu kelapa perlu diperhatikan. Pemrosesan gambar dengan teknik pengelompokan piksel dapat digunakan untuk mendapatkan gambar objek yang diwakili oleh fitur kelas atau material seperti gambar kayu kelapa. Data pertama yang digunakan adalah gambar digital melintang potongan kayu kelapa yang kemudian dikelompokkan berdasarkan fitur-fiturnya menggunakan metode segmentasi citra.

ISBN 978-623-455-329-1

penapersada

PENGOLAHAN CITRA DIGITAL IMPLEMENTASI TEKSTUR KAYU KELAPA DENGAN PYTHON3

Finki Dona Marleny, Okk

PENGOLAHAN CITRA DIGITAL

IMPLEMENTASI TEKSTUR KAYU KELAPA DENGAN PYTHON3

Finki Dona Marleny, Hartini, Ayu Ahadi Ningrum,
Ihdalhubbi Maulida, Rudy Ansari, Mambang

penapersada

PENGOLAHAN CITRA DIGITAL IMPLEMENTASI TEKSTUR KAYU KELAPA DENGAN PYTHON3

Finki Dona Marleny, Hartini, Ayu Ahadi Ningrum, Ihdalhubbi Maulida, Rudy Ansari, Mambang

PT. PENA PERSADA KERTA UTAMA

PENGOLAHAN CITRA DIGITAL IMPLEMENTASI TEKSTUR KAYU KELAPA DENGAN PYTHON3

Penulis:
Finki Dona Marleny, Hartini, Ayu Ahadi Ningrum,
Ihdalhubbi Maulida, Rudy Ansari, Mambang

ISBN: 978-623-455-329-1

Design Cover:
Retnani Nur Briliant

Layout:
Eka Safitry

PT. Pena Persada Kerta Utama
Redaksi:
Jl. Gerilya No. 292 Purwokerto Selatan, Kab. Banyumas
Jawa Tengah. Email: penerbit.penapersada@gmail.com
Website: penapersada.id. Phone: (0281) 7771388

Anggota IKAPI: 178/JTE/2019

Cetakan pertama: 2022

KATA PENGANTAR

Segala puji dan syukur tim penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas segala rahmat, karunia dan kesempatan yang telah diberikan sehingga tim penulis dapat menyelesaikan dan menyusun buku ini. Buku ini membahas tentang pengolahan citra digital, dimana proses segmentasi citra untuk identifikasi tekstur kayu kelapa di implementasikan menggunakan python3. Buku ini dirancang sebagai bahan referensi untuk para mahasiswa, dosen atau siapa saja yang tertarik pada topik segmentasi citra digital. Buku ini lebih menekankan pada cara untuk mengaplikasikan metode segmentasi pada tekstur kayu kelapa.

Ucapan terimakasih tim penulis sampaikan kepada rekan-rekan dan berbagai pihak yang telah mendukung kelancaran dalam penyusunan buku ini. Kritik dan saran penulis terima sebagai bahan masukan penulis untuk terus menulis dan membagikan pengetahuan. Semoga buku ini dapat bermanfaat bagi para pembaca.

Banjarmasin, 2022

ABSTRAK

Kayu kelapa memiliki banyak manfaat dan dapat digunakan sebagai produk bernilai tinggi. Pada pembahasan di buku ini bertujuan untuk mengidentifikasi tekstur kayu kelapa dengan kepadatan tinggi dan sedang. Metode yang digunakan adalah dengan menggunakan segmentasi gambar ambang batas bertingkat yang kemudian di lanjutkan dengan menggunakan metode Gray Level Co-Occurrence Matrix (GLCM). Dalam hal ini, kualitas kayu kelapa perlu diperhatikan. Pemrosesan gambar dengan teknik pengelompokan piksel dapat digunakan untuk mendapatkan gambar objek yang diwakili oleh fitur kelas atau material seperti gambar kayu kelapa. Data pertama yang digunakan adalah gambar digital melintang potongan kayu kelapa yang kemudian dikelompokkan berdasarkan fitur-fiturnya menggunakan metode segmentasi citra.

Kata Kunci: GLCM, Segmentasi, Citra digital, Tekstur kayu

DAFTAR ISI

PENGOLAHAN CITRA DIGITAL
IMPLEMENTASI TEKSTUR KAYU KELAPA
DENGAN PYTHON3

BAB I
PENDAHULUAN

Pohon kelapa banyak tersedia di wilayah Kalimantan Selatan. Kayu kelapa dapat digunakan secara efektif sebagai pengganti kayu konvensional, terutama dalam penggunaannya sebagai bagian bangunan, furnitur, dan barang kerajinan tangan. Bahan kayu kelapa ini merupakan salah satu komoditas ekspor Indonesia ke mancanegara. Di wilayah Kalimantan Selatan, kayu kelapa banyak digunakan sebagai komponen bahan bangunan rumah. Sebab, mayoritas rumah di wilayah Kalimantan Selatan terbuat dari kayu. Kayu kelapa diolah menjadi papan dan dijadikan dinding di beberapa rumah masyarakat Banjar karena harganya yang relatif murah.

Tekstur kayu kelapa memiliki tekstur kayu yang tidak halus dengan tingkat kepadatan tinggi dan sedang. Kayu kelapa lebih diolah secara fisik seperti pembuatan furnitur, komponen rumah, barang kerajinan. Selain itu, juga dapat digunakan sebagai pembuatan arang, briket arang, dan kertas.

Kayu kelapa tidak memiliki mata kayu sehingga proses pemotongan memiliki tingkat kesulitan karena kepadatan serat pada kayu kelapa. Kulit batang kelapa tidak mengelupas, sedangkan yang digunakan adalah bagian dalam batang kelapa. Bagian dalam batang kelapa memiliki cacat distorsi yang lebih besar daripada bagian luarnya. Kayu kelapa memiliki daya tahan alami yang rendah.

Untuk mengidentifikasi tekstur kayu kelapa, diperlukan metode untuk dapat mengenali pola serat kayu kelapa. Untuk mengidentifikasi tekstur kayu kelapa dapat melalui proses segmentasi citra. Segmentasi citra ambang batas bertingkat adalah tahap proses dalam pengenalan pola objek. Pada tahapan pemrosesan citra digital banyak teknik

yang dapat digunakan. Masalah ini dapat diatasi dengan menerapkan sistem pemrosesan citra digital dengan teknik pengelompokan piksel untuk mendapatkan gambar objek yang diwakili oleh fitur, kelas, atau materi. Proses pengelompokan piksel ini masuk kedalam tahapan klasifikasi. Klasifikasi antar objek merupakan tugas yang mudah bagi manusia namun telah terbukti menjadi masalah yang rumit bagi mesin. Sistem klasifikasi terdiri dari database yang berisi pola yang telah ditentukan yang dibandingkan dengan objek yang terdeteksi untuk diklasifikasikan ke dalam kategori yang tepat.

Konsep klasifikasi gambar adalah proses pemetaan angka ke simbol. Untuk mengklasifikasikan satu set data ke dalam kelas atau kategori yang berbeda, perlu dipahami dengan baik hubungan antara data dan kelas-kelas yang diklasifikasikan sebagai. Untuk mencapai hal tersebut dengan menggunakan komputer, komputer harus dilatih. Pelatihan adalah kunci keberhasilan klasifikasi. Teknik klasifikasi awalnya dikembangkan dari penelitian di Bidang Pengenalan Pola.

Pemrosesan gambar dapat digunakan untuk mengidentifikasi jenis kayu berdasarkan teksturnya. Dalam pembahasan ini, analisis tekstur digunakan untuk memberikan model klasifikasi dengan nilai-nilai unik yang mewakili tekstur permukaan kayu kelapa. Tekstur adalah isyarat visual yang penting. Ini adalah fitur yang tersebar luas dalam gambar dan sulit untuk dijelaskan. Ekstraksi fitur tekstur adalah salah satu topik hangat dalam visi komputer, pemrosesan gambar, analisis gambar, dan pengambilan gambar. Metode ekstraksi fitur yang digunakan adalah *Gray Level Co-Occurrence Matrix* (GLCM). GLCM adalah metode statistik untuk memeriksa tekstur yang memperhitungkan hubungan spasial piksel. Fungsi skala abu-abu matriks menunjukkan karakteristik tekstur gambar dengan menentukan seberapa sering pasangan piksel dari nilai tertentu terjadi dalam gambar.

Metode yang akan digunakan didalam pembahasan ini adalah dengan menggunakan metode eksperimen. Dimana citra tektur kayu kelapa akan melalui beberapa tahapan dalam proses pengolahan citra digital. Pada tahapan data awal berupa akuisisi citra digital yaitu mengambil citra dari potongan kayu kelapa secara melintang dan membujur. Kemudian diproses menggunakan segmentasi citra tekstur kayu kelapa secara visual. Untuk mendapatkan kualitas gambar yang baik maka gambar-gambar tersebut diolah melalui pengolahan citra secara komputer visi. Data dari gambar potongan kayu kelapa tersebut kemudian diproses untuk mengubah citra berderajat keabuan menjadi citra biner atau hitam putih sehingga dapat diketahui daerah mana yang termasuk objek dan background dari citra secara jelas.

BAB 2
PENGOLAHAN CITRA DIGITAL

A. Pengolahan Citra Digital

Pengolahan Citra Digital adalah suatu bidang ilmu yang mempelajari suatu citra dibentuk, diolah, dan dianalisis sehingga menghasilkan informasi dan nilai yang dapat dipahami oleh manusia. Pengolahan Citra Digital merupakan usaha untuk melakukan transformasi suatu citra atau gambar menjadi citra lain dengan menggunakan teknik tertentu.

Berikut adalah listing program contoh dasar teknik pengolahan citra untuk memanipulasi saluran warna RGB ke skala abu-abu.

```
import matplotlib.pyplot as plt

from skimage import data
from skimage.color import rgb2gray

original = data.astronaut()
grayscale = rgb2gray(original)

fig, axes = plt.subplots(1, 2, figsize=(8, 4))
ax = axes.ravel()

ax[0].imshow(original)
ax[0].set_title("Original")
ax[1].imshow(grayscale, cmap=plt.cm.gray)
ax[1].set_title("Grayscale")

fig.tight_layout()
plt.show()
```

Dibawah ini adalah listing program teknik pengolahan citra untuk memanipulasi saluran warna RGB ke HSV

```
import matplotlib.pyplot as plt

from skimage import data
from skimage.color import rgb2hsv

rgb_img = data.coffee()
hsv_img = rgb2hsv(rgb_img)
hue_img = hsv_img[:, :, 0]
value_img = hsv_img[:, :, 2]

fig, (ax0, ax1, ax2) = plt.subplots(ncols=3, figsize=(8, 2))

ax0.imshow(rgb_img)
ax0.set_title("RGB image")
ax0.axis('off')
ax1.imshow(hue_img, cmap='hsv')
ax1.set_title("Hue channel")
ax1.axis('off')
ax2.imshow(value_img)
ax2.set_title("Value channel")
ax2.axis('off')

fig.tight_layout()
hue_threshold = 0.04
binary_img = hue_img > hue_threshold

fig, (ax0, ax1) = plt.subplots(ncols=2, figsize=(8, 3))

ax0.hist(hue_img.ravel(), 512)
ax0.set_title("Histogram of the Hue channel with threshold")
ax0.axvline(x=hue_threshold, color='r', linestyle='dashed', linewidth=2)
```

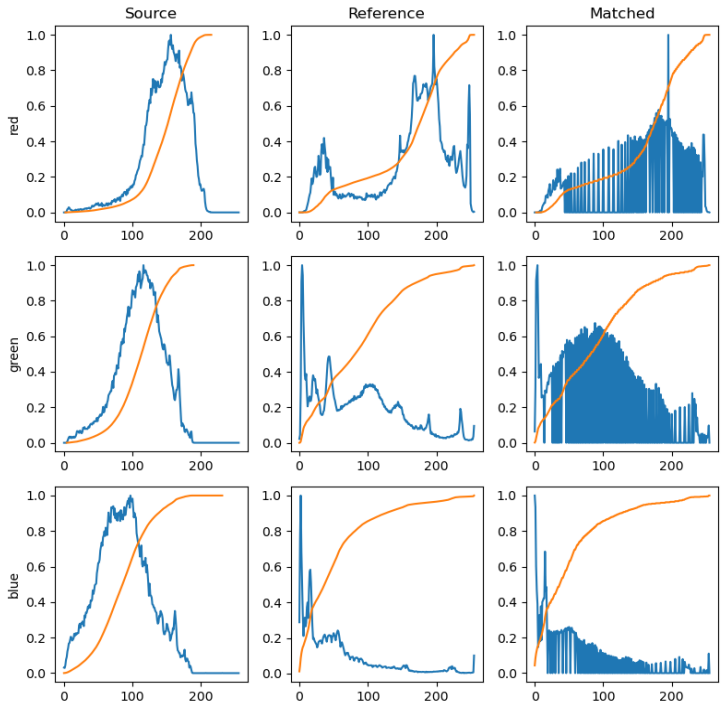
```
ax0.set_xbound(0, 0.12)
ax1.imshow(binary_img)
ax1.set_title("Hue-thresholded image")
ax1.axis('off')

fig.tight_layout()

fig, ax0 = plt.subplots(figsize=(4, 3))

value_threshold = 0.10
binary_img = (hue_img > hue_threshold) | (value_img < value_threshold)

ax0.imshow(binary_img)
ax0.set_title("Hue and value thresholded image")
ax0.axis('off')

fig.tight_layout()
plt.show()
```

Berikut ini adalah listing program teknik pengolahan citra untuk pencocokan histogram.

```
import matplotlib.pyplot as plt

from skimage import data
from skimage import exposure
from skimage.exposure import match_histograms

reference = data.coffee()
image = data.chelsea()

matched = match_histograms(image, reference, channel_axis=-1)

fig, (ax1, ax2, ax3) = plt.subplots(nrows=1, ncols=3,
```

```
figsize=(8, 3),
                              sharex=True, sharey=True)
for aa in (ax1, ax2, ax3):
   aa.set_axis_off()

ax1.imshow(image)
ax1.set_title('Source')
ax2.imshow(reference)
ax2.set_title('Reference')
ax3.imshow(matched)
ax3.set_title('Matched')

plt.tight_layout()
plt.show()

fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(8, 8))


for i, img in enumerate((image, reference, matched)):
   for c, c_color in enumerate(('red', 'green', 'blue')):
      img_hist, bins = exposure.histogram(img[..., c],
source_range='dtype')
      axes[c, i].plot(bins, img_hist / img_hist.max())
      img_cdf, bins =
exposure.cumulative_distribution(img[..., c])
      axes[c, i].plot(bins, img_cdf)
      axes[c, 0].set_ylabel(c_color)

axes[0, 0].set_title('Source')
axes[0, 1].set_title('Reference')
axes[0, 2].set_title('Matched')

plt.tight_layout()
plt.show()
```

Hasil Histogram citra:



Gambar 1. Histogram Citra

B. Segmentasi Citra

Segmentasi citra merupakan tahapan penting dalam proses pengenalan pola. Dari objek yang berhasil tersegmentasi, maka selanjutnya dilakukan proses ekstraksi. Segmentasi citra merupakan proses yang ditujukan untuk mendapatkan objek-objek yang terkandung di dalam citra atau membagi citra ke dalam beberapa objek berdasarkan kemiripan dari objek tersebut. Pada citra yang mengandung hanya satu objek, objek dibedakan dari latar belakangnya.

Tujuan segmentasi citra adalah untuk mempartisi gambar menjadi beberapa wilayah yang tidak tumpang tindih dengan karakteristik yang homogen, seperti intensitas, warna, dan tekstur.

Berikut ini adalah contoh dasar teknik segmentasi citra menggunakan compact watershed

```
import numpy as np
from skimage import data, util, filters, color
from skimage.segmentation import watershed
import matplotlib.pyplot as plt

coins = data.coins()
edges = filters.sobel(coins)

grid = util.regular_grid(coins.shape, n_points=468)

seeds = np.zeros(coins.shape, dtype=int)
seeds[grid] =
np.arange(seeds[grid].size).reshape(seeds[grid].shape) + 1

w0 = watershed(edges, seeds)
w1 = watershed(edges, seeds, compactness=0.01)

fig, (ax0, ax1) = plt.subplots(1, 2)

ax0.imshow(color.label2rgb(w0, coins, bg_label=-1))
ax0.set_title('Classical watershed')

ax1.imshow(color.label2rgb(w1, coins, bg_label=-1))
ax1.set_title('Compact watershed')

plt.show()
```

Berikut ini adalah contoh dasar teknik segmentasi citra memperluas label segmentasi tanpa tumpang tindih.

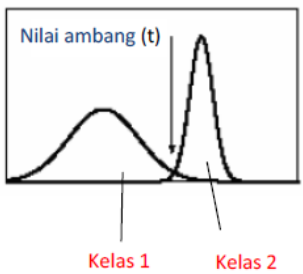
```
import numpy as np
import matplotlib.pyplot as plt

from skimage.filters import sobel
from skimage.measure import label
from skimage.segmentation import watershed, expand_labels
from skimage.color import label2rgb
from skimage import data

coins = data.coins()

# Make segmentation using edge-detection and watershed.
edges = sobel(coins)

# Identify some background and foreground pixels from the intensity values.
# These pixels are used as seeds for watershed.
markers = np.zeros_like(coins)
foreground, background = 1, 2
markers[coins < 30.0] = background
markers[coins > 150.0] = foreground

ws = watershed(edges, markers)
seg1 = label(ws == foreground)

expanded = expand_labels(seg1, distance=10)

# Show the segmentations.
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(9, 5),
                         sharex=True, sharey=True)

color1 = label2rgb(seg1, image=coins, bg_label=0)
```

```
axes[0].imshow(color1)
axes[0].set_title('Sobel+Watershed')

color2 = label2rgb(expanded, image=coins, bg_label=0)
axes[1].imshow(color2)
axes[1].set_title('Expanded labels')

for a in axes:
   a.axis('off')
fig.tight_layout()
plt.show()
```

C. Thresholding Metode Otsu

Dalam pengolahan citra digital, metode Otsu digunakan untuk pengambangan (thresholding) citra otomatis. Algoritma ini mengembalikan nilai ambang intensitas tunggal yang membagi piksel-piksel menjadi dua kelas, yaitu latar depan dan latar belakang. Nilai ambang ini ditentukan dengan meminimalkan ragam intensitas dalam kelas atau memaksimalkan ragam intensitas antarkelas.

Metode Otsu dipublikasikan oleh Nobuyuki Otsu pada tahun 1979. Metode ini menentukan nilai ambang dengan cara membedakan dua kelompok, yaitu objek dan latar belakang, yang memiliki bagian yang saling bertumpukan, berdasarkan histogram.



Gambar 2. Penentuan nilai ambang untuk memperoleh hasil yang optimal

Metode Otsu cukup baik jika histogram dianggap memiliki persebaran dua puncak serta memiliki lembah yang curam dan dalam di antara dua puncak. Namun, bila luas objek (latar depan) berukuran kecil dibandingkan luas latar belakang, histogram tidak lagi memiliki sifat dua puncak. Jika ragam objek dan ragam latar belakang cukup besar dibanding selisih rata-rata atau jika citra sangat rusak akibat derau aditif, kecuraman lembah pada histogram menurun. Akibatnya, nilai ambang yang dihasilkan menyebabkan kesalahan segmentasi.

Multilevel Thresholding

Pada peng-ambangan beraras-jamak (multilevel thresholding), citra dibagi menjadi beberapa bagian dengan menggunakan beberapa nilai ambang.

Adaptive Thresholding

Peng-ambangan adaptif (adaptive thresholding) merupakan peng-ambangan yang menggunakan nilai ambang lokal, yang dihitung secara adaptif berdasarkan statistika piksel-piksel tetangga. Hal ini didasarkan kenyataan bahwa bagian-bagian kecil dalam citra mempunyai iluminasi yang sama, sehingga lebih tepat kalau nilai ambang dihitung berdasarkan bagian-bagian kecil dalam citra dan bukan berdasarkan seluruh piksel dalam citra.

Nilai threshold ditentukan berdasarkan nilai piksel tetangga pada "window" dengan ukuran tertentu. Nilai threshold untuk setiap "window" dapat berbeda-beda (adaptive).

Algoritma penentuan threshold:

1. Rerata (mean)
2. Max-min
3. Nilai tengah (median)

Berikut ini adalah contoh dasar teknik segmentasi citra dengan metode otsu.

```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

from skimage import data
from skimage.filters import threshold_multiotsu

# Setting the font size for all plots.
matplotlib.rcParams['font.size'] = 9

# The input image.
image = data.camera()

# Applying multi-Otsu threshold for the default value, generating
# three classes.
thresholds = threshold_multiotsu(image)

# Using the threshold values, we generate the three regions.
regions = np.digitize(image, bins=thresholds)

fig, ax = plt.subplots(nrows=1, ncols=3, figsize=(10, 3.5))

# Plotting the original image.
ax[0].imshow(image, cmap='gray')
ax[0].set_title('Original')
ax[0].axis('off')

# Plotting the histogram and the two thresholds obtained from
# multi-Otsu.
ax[1].hist(image.ravel(), bins=255)
```

```
ax[1].set_title('Histogram')
for thresh in thresholds:
    ax[1].axvline(thresh, color='r')

# Plotting the Multi Otsu result.
ax[2].imshow(regions, cmap='jet')
ax[2].set_title('Multi-Otsu result')
ax[2].axis('off')

plt.subplots_adjust()

plt.show()
```

BAB 3
KLASIFIKASI CITRA

A. Proses Klasifikasi

Klasifikasi Citra Konsep klasifikasi gambar adalah proses pemetaan angka ke symbol [6]. Untuk mengklasifikasikan seperangkat data ke dalam kelas atau kategori yang berbeda, hubungan antara data dan kelas yang diklasifikasikan harus dipahami dengan baik. Untuk mencapai hal tersebut menggunakan komputer maka komputer harus dilatih. Pelatihan adalah kunci keberhasilan klasifikasi. Teknik klasifikasi awalnya dikembangkan dari penelitian di Pattern Recognitionfield [7]. Pengklasifikasi memiliki keuntungan dari analis atau pengetahuan domain yang dengannya pemandu dapat dipandu untuk mempelajari hubungan antara data dan kelas. Jumlah kelas, piksel prototipe untuk setiap kelas dapat diidentifikasi menggunakan pengetahuan sebelumnya.

B. Proses Klasifikasi

Proses klasifikasi terdiri dari langkah-langkah berikut:

1. Pre-processing Pada langkah ini terjadi pemrosesan atau koreksi atmosfer gambar, penghapusan noise, transformasi gambar, analisis komponen utama dan lain-lain.
2. Deteksi dan ekstraksi objek Deteksi mencakup deteksi posisi dan karakteristik lain dari objek gambar yang diperoleh dari kamera. Dan dalam ekstraksi, dari objek pada gambar terdeteksi memperkirakan lintasan objek di bidang gambar.
3. Training Pada langkah training data, pemilihan atribut tertentu yang paling menggambarkan pola.

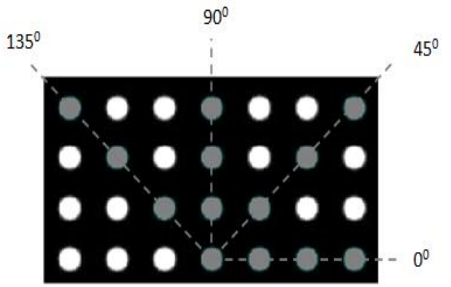
4. Klasifikasi objek Pada langkah klasifikasi objek mengategorikan objek yang terdeteksi ke dalam kelas yang telah ditentukan dengan menggunakan metode yang sesuai yang membandingkan pola gambar dengan pola target.

C. Metode GLCM

GLCM pertama kali diusulkan oleh Haralick (1973) yang terdiri dari 28 fitur untuk menjelaskan pola spasial (Kulkarni, 1994). GLCM menggunakan perhitungan tekstur pada orde kedua. Pengukuran tekstur pada orde pertama menggunakan perhitungan statistika didasarkan pada nilai piksel citra asli.

GLCM terdiri dari 4 arah:

1. 00^0
2. 450^0
3. 900^0
4. 1350^0



GLCM terdiri dari 28 fitur, namun yang sering digunakan hanya 5 (fitur):

1. ASM → ukuran homogenitas citra
2. Contrast → ukuran keberadaan variasi aras keabuan piksel citra
3. Inverse Different Moment (IDM) → untuk mengukur homogenitas
4. Entropi → ukuran ketidakteraturan aras keabuan di dalam citra
5. Korelasi → ukuran ketergantungan linear antarnilai aras keabuan dalam citra

Berikut ini adalah contoh listing program menggunakan metode GLCM.

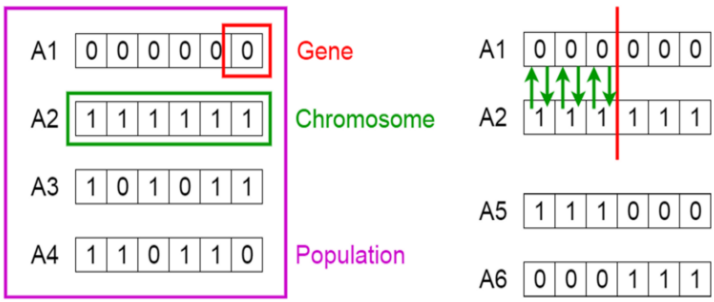
```
import matplotlib.pyplot as plt

from skimage.feature import graycomatrix, graycoprops
from skimage import data


PATCH_SIZE = 21

# open the camera image
image = data.camera()

# select some patches from grassy areas of the image
grass_locations = [(280, 454), (342, 223), (444, 192), (455, 455)]
grass_patches = []
for loc in grass_locations:
    grass_patches.append(image[loc[0]:loc[0] + PATCH_SIZE,
                               loc[1]:loc[1] + PATCH_SIZE])

# select some patches from sky areas of the image
sky_locations = [(38, 34), (139, 28), (37, 437), (145, 379)]
sky_patches = []
for loc in sky_locations:
    sky_patches.append(image[loc[0]:loc[0] + PATCH_SIZE,
                             loc[1]:loc[1] + PATCH_SIZE])

# compute some GLCM properties each patch
xs = []
ys = []
for patch in (grass_patches + sky_patches):
    glcm = graycomatrix(patch, distances=[5], angles=[0], levels=256,
```

```
                symmetric=True, normed=True)
    xs.append(graycoprops(glcm, 'dissimilarity')[0, 0])
    ys.append(graycoprops(glcm, 'correlation')[0, 0])

# create the figure
fig = plt.figure(figsize=(8, 8))

# display original image with locations of patches
ax = fig.add_subplot(3, 2, 1)
ax.imshow(image, cmap=plt.cm.gray,
          vmin=0, vmax=255)
for (y, x) in grass_locations:
    ax.plot(x + PATCH_SIZE / 2, y + PATCH_SIZE / 2, 'gs')
for (y, x) in sky_locations:
    ax.plot(x + PATCH_SIZE / 2, y + PATCH_SIZE / 2, 'bs')
ax.set_xlabel('Original Image')
ax.set_xticks([])
ax.set_yticks([])
ax.axis('image')

# for each patch, plot (dissimilarity, correlation)
ax = fig.add_subplot(3, 2, 2)
ax.plot(xs[:len(grass_patches)], ys[:len(grass_patches)], 'go',
        label='Grass')
ax.plot(xs[len(grass_patches):], ys[len(grass_patches):], 'bo',
        label='Sky')
ax.set_xlabel('GLCM Dissimilarity')
ax.set_ylabel('GLCM Correlation')
ax.legend()

# display the image patches
for i, patch in enumerate(grass_patches):
    ax = fig.add_subplot(3, len(grass_patches),
len(grass_patches)*1 + i + 1)
    ax.imshow(patch, cmap=plt.cm.gray,
              vmin=0, vmax=255)
```

```
    ax.set_xlabel('Grass %d' % (i + 1))

for i, patch in enumerate(sky_patches):
    ax = fig.add_subplot(3, len(sky_patches),
len(sky_patches)*2 + i + 1)
    ax.imshow(patch, cmap=plt.cm.gray,
              vmin=0, vmax=255)
    ax.set_xlabel('Sky %d' % (i + 1))


# display the patches and plot
fig.suptitle('Grey level co-occurrence matrix features',
fontsize=14, y=1.05)
plt.tight_layout()
plt.show()
```

D. Algoritma Genetika

Algoritma genetika adalah heuristik pencarian yang terinspirasi oleh teori evolusi alami Charles Darwin. Algoritma ini mencerminkan proses seleksi alam di mana individu yang paling cocok dipilih untuk reproduksi untuk menghasilkan keturunan dari generasi berikutnya.



Pengertian Seleksi Alam

Proses seleksi alam dimulai dengan pemilihan individu yang paling cocok dari suatu populasi. Mereka menghasilkan keturunan yang mewarisi karakteristik orang tua dan akan ditambahkan ke generasi berikutnya. Jika orang tua memiliki kebugaran yang lebih baik, keturunannya akan lebih baik daripada orang tua dan memiliki peluang lebih baik untuk bertahan hidup. Proses ini terus berulang dan pada akhirnya akan ditemukan generasi dengan individu yang paling cocok.

Gagasan ini dapat diterapkan untuk masalah pencarian. Kami mempertimbangkan satu set solusi untuk masalah dan memilih set yang terbaik dari mereka.

Lima fase dipertimbangkan dalam algoritma genetika.

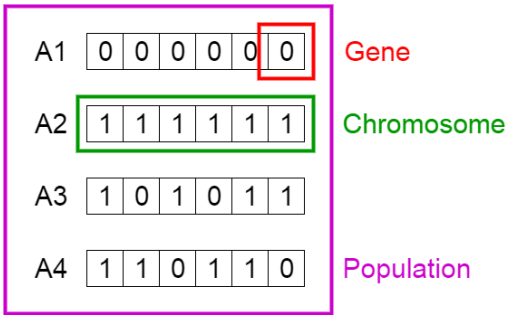
1. Populasi awal
2. Fungsi kebugaran
3. Pilihan
4. pindah silang
5. Mutasi

Populasi Awal

Prosesnya dimulai dengan sekumpulan individu yang disebut Populasi . Setiap individu adalah solusi untuk masalah yang ingin Anda pecahkan.

Seorang individu dicirikan oleh seperangkat parameter (variabel) yang dikenal sebagai Gen. Gen bergabung menjadi string untuk membentuk Kromosom (solusi).

Dalam algoritma genetika, himpunan gen dari suatu individu direpresentasikan menggunakan string, dalam bentuk alfabet. Biasanya, nilai biner digunakan (string 1s dan 0s). Kami mengatakan bahwa kami mengkodekan gen dalam kromosom**.**



Populasi, Kromosom dan Gen

Fungsi fitness

Fungsi fitness menentukan seberapa fit seseorang (kemampuan individu untuk bersaing dengan individu lain). Ini memberikan skor fitness untuk setiap individu. Probabilitas bahwa seorang individu akan dipilih untuk reproduksi didasarkan pada skor kebugarannya.

fase seleksi

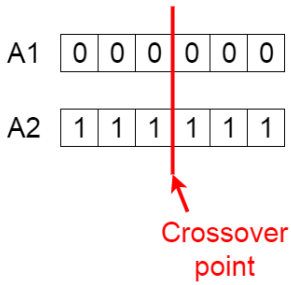
Ide dari fase seleksi adalah untuk memilih individu yang paling cocok dan membiarkan mereka mewariskan gen mereka ke generasi berikutnya.

Dua pasang individu (orang tua) dipilih berdasarkan skor kebugaran mereka. Individu dengan fitness tinggi memiliki peluang lebih besar untuk dipilih untuk reproduksi.

Crossover

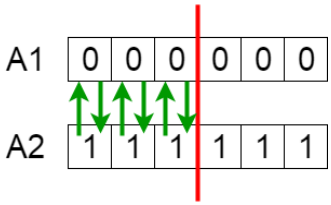
Crossover adalah fase paling signifikan dalam algoritma genetika. Untuk setiap pasangan tetua yang akan dikawinkan, titik persilangan dipilih secara acak dari dalam gen.

Misalnya, pertimbangkan titik crossover menjadi 3 seperti yang ditunjukkan di bawah ini.



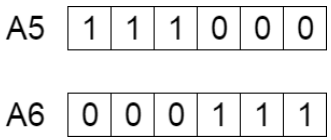
Titik persimpangan

Keturunan dibuat dengan menukar gen orang tua di antara mereka sendiri sampai titik crossover tercapai.



Pertukaran gen di antara orang tua

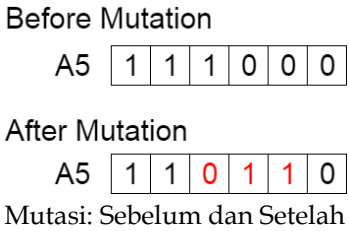
Keturunan baru ditambahkan ke populasi.



Keturunan baru

Mutasi

Pada keturunan baru tertentu yang terbentuk, beberapa gen mereka dapat mengalami mutasi dengan probabilitas acak yang rendah. Ini menyiratkan bahwa beberapa bit dalam string bit dapat dibalik.



Mutasi: Sebelum dan Setelah

Mutasi terjadi untuk menjaga keragaman dalam populasi dan mencegah konvergensi dini.

Penghentian

Algoritma berhenti jika populasi telah konvergen (tidak menghasilkan keturunan yang berbeda nyata dengan generasi sebelumnya). Kemudian dikatakan bahwa algoritma genetika telah menyediakan satu set solusi untuk masalah kita.

Berikut adalah contoh listing program menggunakan Algoritma genetik.

```
import numpy
import ga

"""
The y=target is to maximize this equation ASAP:
    y = w1x1+w2x2+w3x3+w4x4+w5x5+6wx6
    where (x1,x2,x3,x4,x5,x6)=(4,-2,3.5,5,-11,-4.7)
    What are the best values for the 6 weights w1 to w6?
    We are going to use the genetic algorithm for the best possible values after a number of generations.
"""

# Inputs of the equation.
equation_inputs = [4,-2,3.5,5,-11,-4.7]

# Number of the weights we are looking to optimize.
num_weights = 6
```

```
"""
Genetic algorithm parameters:
    Mating pool size
    Population size
"""
sol_per_pop = 8
num_parents_mating = 4

# Defining the population size.
pop_size = (sol_per_pop,num_weights) # The population will have sol_per_pop chromosome where each chromosome has num_weights genes.
#Creating the initial population.
new_population = numpy.random.uniform(low=-4.0, high=4.0, size=pop_size)
print(new_population)

num_generations = 5
for generation in range(num_generations):
    print("Generation : ", generation)
    # Measing the fitness of each chromosome in the population.
    fitness = ga.cal_pop_fitness(equation_inputs, new_population)

    # Selecting the best parents in the population for mating.
    parents = ga.select_mating_pool(new_population, fitness,
                                    num_parents_mating)

    # Generating next generation using crossover.
    offspring_crossover = ga.crossover(parents,
                                       offspring_size=(pop_size[0]-parents.shape[0], num_weights))

    # Adding some variations to the offsrping using
```

```
mutation.
    offspring_mutation = ga.mutation(offspring_crossover)

    # Creating the new population based on the parents and offspring.
    new_population[0:parents.shape[0], :] = parents
    new_population[parents.shape[0]:, :] = offspring_mutation

    # The best result in the current iteration.
    print("Best result : ", numpy.max(numpy.sum(new_population*equation_inputs, axis=1)))

# Getting the best solution after iterating finishing all generations.
#At first, the fitness is calculated for each solution in the final generation.
fitness = ga.cal_pop_fitness(equation_inputs, new_population)
# Then return the index of that solution corresponding to the best fitness.
best_match_idx = numpy.where(fitness == numpy.max(fitness))

print("Best solution : ", new_population[best_match_idx, :])
print("Best solution fitness : ", fitness[best_match_idx])
```

```
import numpy

# This project is extended and a library called PyGAD is released to build the genetic algorithm.
# PyGAD documentation: https://pygad.readthedocs.io
# Install PyGAD: pip install pygad
# PyGAD source code at GitHub: https://github.com/ahmedfgad/GeneticAlgorithmPython
```

```
def cal_pop_fitness(equation_inputs, pop):
    # Calculating the fitness value of each solution in the current population.
    # The fitness function caulcuates the sum of products between each input and its corresponding weight.
    fitness = numpy.sum(pop*equation_inputs, axis=1)
    return fitness

def select_mating_pool(pop, fitness, num_parents):
    # Selecting the best individuals in the current generation as parents for producing the offspring of the next generation.
    parents = numpy.empty((num_parents, pop.shape[1]))
    for parent_num in range(num_parents):
        max_fitness_idx = numpy.where(fitness == numpy.max(fitness))
        max_fitness_idx = max_fitness_idx[0][0]
        parents[parent_num, :] = pop[max_fitness_idx, :]
        fitness[max_fitness_idx] = -99999999999
    return parents

def crossover(parents, offspring_size):
    offspring = numpy.empty(offspring_size)
    # The point at which crossover takes place between two parents. Usually it is at the center.
    crossover_point = numpy.uint8(offspring_size[1]/2)

    for k in range(offspring_size[0]):
        # Index of the first parent to mate.
        parent1_idx = k%parents.shape[0]
        # Index of the second parent to mate.
        parent2_idx = (k+1)%parents.shape[0]
        # The new offspring will have its first half of its genes taken from the first parent.
        offspring[k, 0:crossover_point] = parents[parent1_idx,
```

```
0:crossover_point]
        # The new offspring will have its second half of its genes taken from the second parent.
        offspring[k, crossover_point:] = parents[parent2_idx, crossover_point:]
    return offspring

def mutation(offspring_crossover):
    # Mutation changes a single gene in each offspring randomly.
    for idx in range(offspring_crossover.shape[0]):
        # The random value to be added to the gene.
        random_value = numpy.random.uniform(-1.0, 1.0, 1)
        offspring_crossover[idx, 4] = offspring_crossover[idx, 4] + random_value
    return offspring_crossover
```

BAB 4
PENGOLAHAN DATA CITRA KAYU KELAPA

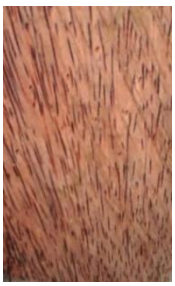
A. Pengolahan data citra

Data awal berupa gambar digital potongan kayu secara melintang dimasukkan dalam database kayu secara visual. Untuk mendapatkan kualitas gambar yang baik maka gambar-gambar tersebut diolah melalui image processing. Database dari gambar potongan kayu tersebut kemudian dilatih untuk penentuan klasifikasi kayu berdasar fitur, fitur ini secara visual berupa kerapatan dari noda-noda yang terdapat pada gambar kayu tersebut. Setelah proses pelatihan dari data tersebut kemudian dilakukan pengujian terhadap visual potongan kayu untuk ditentukan klasifikasinya.

B. Pengolahan data awal

Data yang digunakan adalah data potongan kayu kelapa secara melintang dan membujur. Berikut adalah data citra dari potongan kayu kelapa pada citra (a) dan (b) gambar dibawah adalah citra kayu kelapa dengan potongan membujur. Pada gambar (c) dan (d) adalah potongan kayu kelapa secara melintang.

(a)



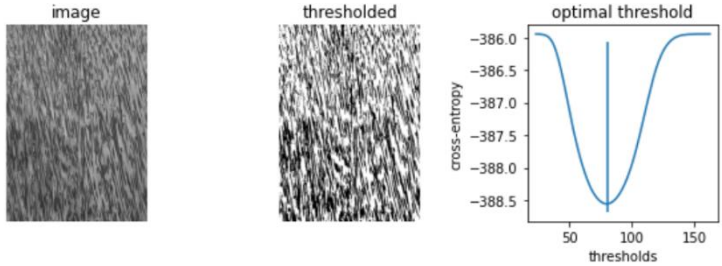
(B)

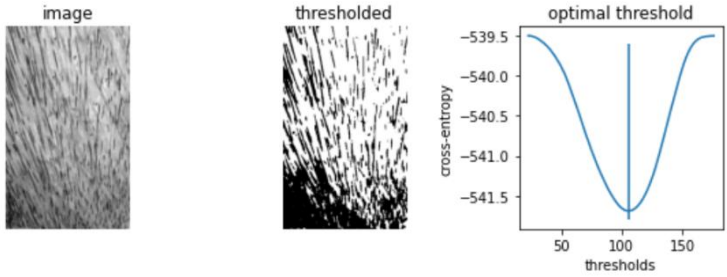
(c) (D)

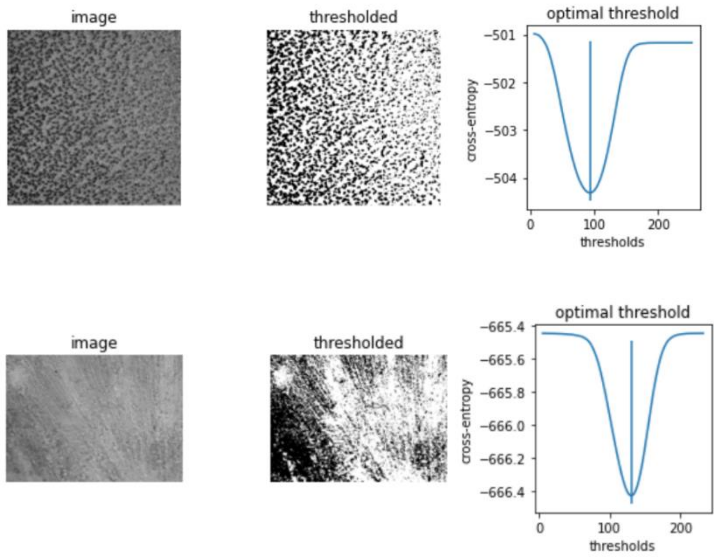
Gambar 3. Potongan kayu kelapa

C. Proses ambang batas citra

Ambang batas potongan kayu kelapa membujur. Pertama, mari kita plot entropi silang untuk gambar di semua ambang batas yang mungkin.



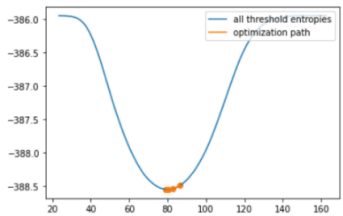




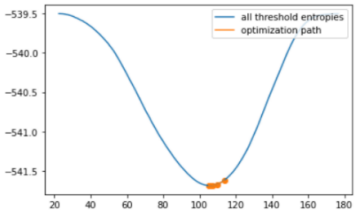
Gambar 4. Segmentasi gambar menghasilkan ambang batas optimal

Tabel 1. Entropi silang untuk gambar pada semua ambang batas

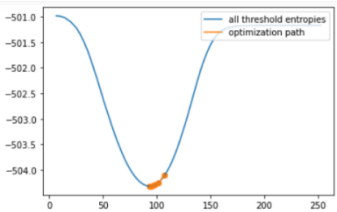
Kode citra	batas optimal	Batas optimal yang dihitung
(a)	80.5	79.05427140704299
(b)	105.5	105.17147645749911
(c)	93.5	93.38627289435556
(d)	130.5	131.25892578323436



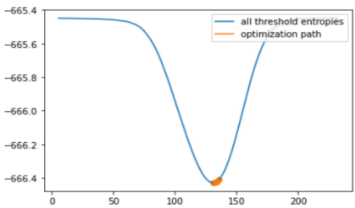
(a) Hanya 6 ambang batas yang diperiksa



(b) Hanya 7 ambang batas yang diperiksa



(c) Hanya 9 ambang batas yang diperiksa.

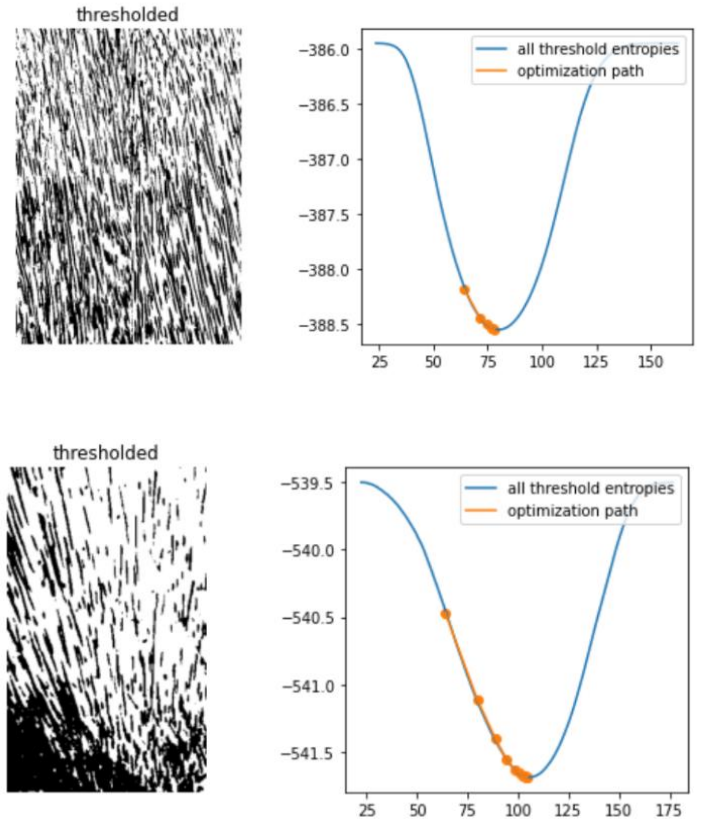


(d) Hanya 6 ambang batas yang diperiksa.

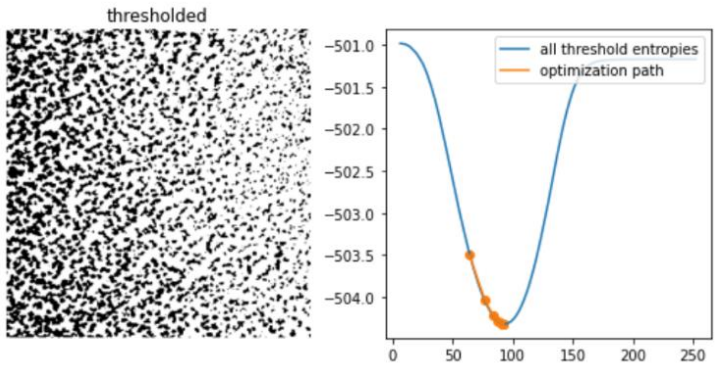
Gambar 5. Proses optimasi

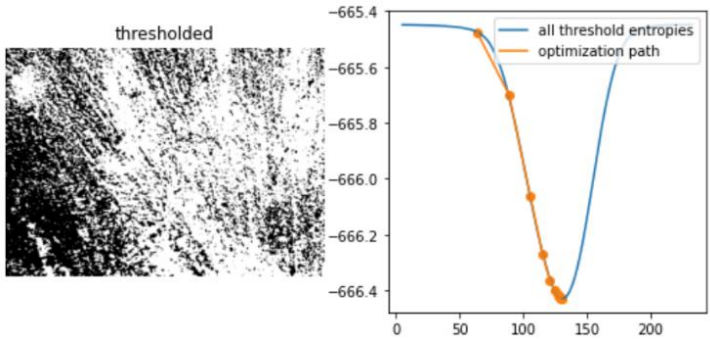
Peoses ini jauh lebih efisien dari pada pendekatan *brute force*. Namun, pada beberapa gambar, cross-entropy tidak *cembung* , artinya memiliki optimum tunggal. Dalam hal ini, penurunan gradien dapat menghasilkan ambang batas yang tidak optimal. Dalam contoh ini, kita melihat bagaimana nilai data awal yang kurang baik untuk pengoptimalan menghasilkan dalam pemilihan ambang batas.

Dalam gambar ini, awal *default* , nilai gambar rata-rata, sebenarnya terletak *tepat* di atas puncak antara dua "lembah" fungsi tujuan.



Gambar 6. Segmentasi gambar yang menghasilkan semua entropi ambang batas

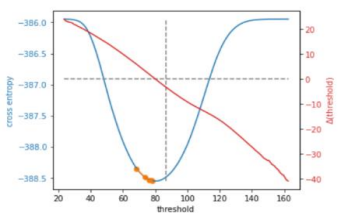




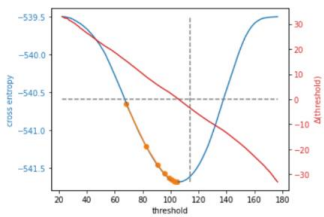
Gambar 7. Segmentasi gambar menghasilkan semua entropi ambang batas

Untuk melihat apa yang terjadi, mari kita definisikan sebuah fungsi, *li_gradient*, yang mereplikasi perulangan dalam dari metode *Li* dan mengembalikan *perubahan* dari nilai ambang saat ini ke yang berikutnya. Ketika gradien ini adalah 0, kita berada pada apa yang disebut *titik stasioner* dan *Li* mengembalikan nilai. Ketika negatif, tebakan ambang berikutnya akan lebih rendah, dan ketika positif, tebakan berikutnya akan lebih tinggi.

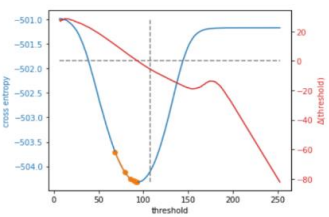
Dalam plot di bawah, pada gambar menunjukkan lintas-entropi dan jalur pembaruan *Li* ketika tebakan awal berada di sisi *kanan* puncak entropi itu. Kami melapisi gradien pembaruan ambang batas, menandai garis gradien 0 dan tebakan awal default dengan *threshold_li*.



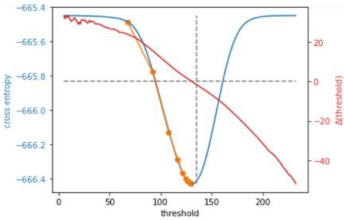
(a) 5 examined, optimum: 78.04613147213259



(b) 11 examined, optimum: 104.58427738739046



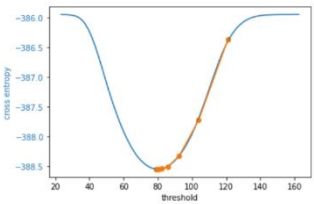
(c) 9 examined, optimum: 92.76400645925322



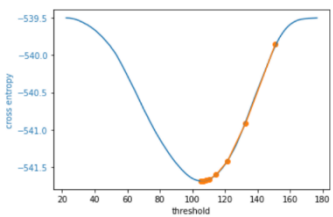
(d) 12 examined, optimum: 129.9348845513801

Gambar 8. Cross entropy

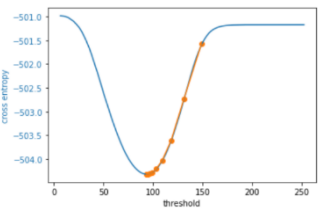
Selain memungkinkan pengguna untuk memberikan angka sebagai tebakan awal, *skimage. filters.threshold_li()* dapat menerima fungsi yang membuat tebakan dari intensitas gambar, seperti yang *numpy.mean()* dilakukan secara default. Ini mungkin pilihan yang baik ketika banyak gambar dengan rentang yang berbeda perlu diproses.



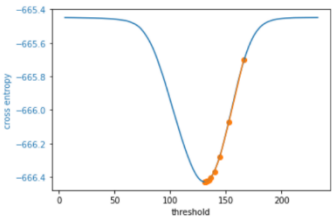
(a) 9 examined, optimum: 79.05427140704299



(b) 10 examined, optimum: 105.17147645749911



(c) 12 examined, optimum: 93.38627289435556



(d) 11 examined, optimum: 131.25892578323436

Listing program ambang batas Li

```
import numpy as np
import matplotlib.pyplot as plt
from skimage import data
from skimage import filters
from skimage.filters.thresholding import _cross_entropy

cell = data.cell()
camera = data.camera()

thresholds = np.arange(np.min(camera) + 1.5,
np.max(camera) - 1.5)
entropies = [_cross_entropy(camera, t) for t in thresholds]

optimal_camera_threshold =
thresholds[np.argmin(entropies)]

fig, ax = plt.subplots(1, 3, figsize=(8, 3))

ax[0].imshow(camera, cmap='gray')
ax[0].set_title('image')
ax[0].set_axis_off()

ax[1].imshow(camera > optimal_camera_threshold,
cmap='gray')
ax[1].set_title('thresholded')
ax[1].set_axis_off()

ax[2].plot(thresholds, entropies)
ax[2].set_xlabel('thresholds')
ax[2].set_ylabel('cross-entropy')
ax[2].vlines(optimal_camera_threshold,
        ymin=np.min(entropies) - 0.05 * np.ptp(entropies),
        ymax=np.max(entropies) - 0.05 * np.ptp(entropies))
ax[2].set_title('optimal threshold')
```

```
fig.tight_layout()

print('The brute force optimal threshold is:', optimal_camera_threshold)
print('The computed optimal threshold is:', filters.threshold_li(camera))

plt.show()
```

```
iter_thresholds = []

optimal_threshold = filters.threshold_li(camera,

iter_callback=iter_thresholds.append)
iter_entropies = [_cross_entropy(camera, t) for t in iter_thresholds]

print('Only', len(iter_thresholds), 'thresholds examined.')

fig, ax = plt.subplots()

ax.plot(thresholds, entropies, label='all threshold entropies')
ax.plot(iter_thresholds, iter_entropies, label='optimization path')
ax.scatter(iter_thresholds, iter_entropies, c='C1')
ax.legend(loc='upper right')

plt.show()
```

```
iter_thresholds2 = []

opt_threshold2 = filters.threshold_li(cell, initial_guess=64,

iter_callback=iter_thresholds2.append)
```

```
thresholds2 = np.arange(np.min(cell) + 1.5, np.max(cell) - 1.5)
entropies2 = [_cross_entropy(cell, t) for t in thresholds]
iter_entropies2 = [_cross_entropy(cell, t) for t in iter_thresholds2]

fig, ax = plt.subplots(1, 3, figsize=(8, 3))

ax[0].imshow(cell, cmap='magma')
ax[0].set_title('image')
ax[0].set_axis_off()

ax[1].imshow(cell > opt_threshold2, cmap='gray')
ax[1].set_title('thresholded')
ax[1].set_axis_off()

ax[2].plot(thresholds2, entropies2, label='all threshold entropies')
ax[2].plot(iter_thresholds2, iter_entropies2, label='optimization path')
ax[2].scatter(iter_thresholds2, iter_entropies2, c='C1')
ax[2].legend(loc='upper right')

plt.show()
```

```
iter_thresholds3 = []

opt_threshold3 = filters.threshold_li(cell,

iter_callback=iter_thresholds3.append)

iter_entropies3 = [_cross_entropy(cell, t) for t in iter_thresholds3]

fig, ax = plt.subplots(1, 3, figsize=(8, 3))
```

```
ax[0].imshow(cell, cmap='magma')
ax[0].set_title('image')
ax[0].set_axis_off()

ax[1].imshow(cell > opt_threshold3, cmap='gray')
ax[1].set_title('thresholded')
ax[1].set_axis_off()

ax[2].plot(thresholds2, entropies2, label='all threshold entropies')
ax[2].plot(iter_thresholds3, iter_entropies3, label='optimization path')
ax[2].scatter(iter_thresholds3, iter_entropies3, c='C1')
ax[2].legend(loc='upper right')

plt.show()
```

```
def li_gradient(image, t):
    """Find the threshold update at a given threshold."""
    foreground = image > t
    mean_fore = np.mean(image[foreground])
    mean_back = np.mean(image[~foreground])
    t_next = ((mean_back - mean_fore) /
              (np.log(mean_back) - np.log(mean_fore)))
    dt = t_next - t
    return dt


iter_thresholds4 = []
opt_threshold4 = filters.threshold_li(cell, initial_guess=68,

iter_callback=iter_thresholds4.append)
iter_entropies4 = [_cross_entropy(cell, t) for t in iter_thresholds4]
print(len(iter_thresholds4), 'examined, optimum:', opt_threshold4)
```

```
gradients = [li_gradient(cell, t) for t in thresholds2]

fig, ax1 = plt.subplots()
ax1.plot(thresholds2, entropies2)
ax1.plot(iter_thresholds4, iter_entropies4)
ax1.scatter(iter_thresholds4, iter_entropies4, c='C1')
ax1.set_xlabel('threshold')
ax1.set_ylabel('cross entropy', color='C0')
ax1.tick_params(axis='y', labelcolor='C0')

ax2 = ax1.twinx()
ax2.plot(thresholds2, gradients, c='C3')
ax2.hlines([0], xmin=thresholds2[0], xmax=thresholds2[-1],
        colors='gray', linestyles='dashed')
ax2.vlines(np.mean(cell),          ymin=np.min(gradients),
ymax=np.max(gradients),
        colors='gray', linestyles='dashed')
ax2.set_ylabel(r'$\Delta$(threshold)', color='C3')
ax2.tick_params(axis='y', labelcolor='C3')

fig.tight_layout()

plt.show()
```

```
def quantile_95(image):
   # you can use np.quantile(image, 0.95) if you have
NumPy>=1.15
   return np.percentile(image, 95)

iter_thresholds5 = []
opt_threshold5          =          filters.threshold_li(cell,
initial_guess=quantile_95,

iter_callback=iter_thresholds5.append)
iter_entropies5 = [_cross_entropy(cell, t) for t in
iter_thresholds5]
print(len(iter_thresholds5), 'examined, optimum:',
```

```
opt_threshold5)

fig, ax1 = plt.subplots()
ax1.plot(thresholds2, entropies2)
ax1.plot(iter_thresholds5, iter_entropies5)
ax1.scatter(iter_thresholds5, iter_entropies5, c='C1')
ax1.set_xlabel('threshold')
ax1.set_ylabel('cross entropy', color='C0')
ax1.tick_params(axis='y', labelcolor='C0')

plt.show()
```

Listing Program fitur GLCM.

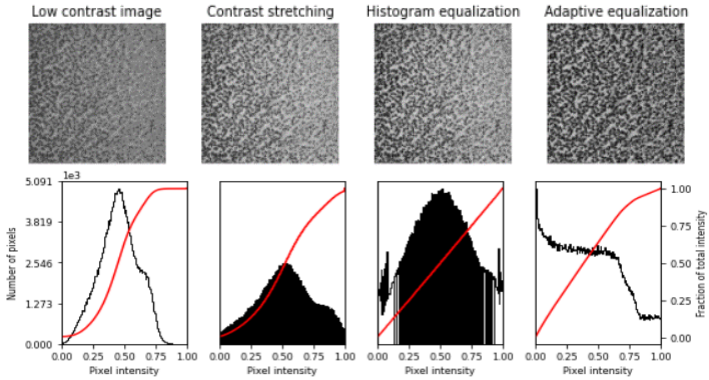
```
import matplotlib.pyplot as plt

from skimage.feature import graycomatrix, graycoprops
from skimage import data


PATCH_SIZE = 21

# open the camera image
image = data.camera()

# select some patches from grassy areas of the image
grass_locations = [(280, 454), (342, 223), (444, 192), (455,
455)]
grass_patches = []
for loc in grass_locations:
    grass_patches.append(image[loc[0]:loc[0]                    +
PATCH_SIZE,
                           loc[1]:loc[1] + PATCH_SIZE])

# select some patches from sky areas of the image
sky_locations = [(38, 34), (139, 28), (37, 437), (145, 379)]
```

```
sky_patches = []
for loc in sky_locations:
   sky_patches.append(image[loc[0]:loc[0] + PATCH_SIZE,
                      loc[1]:loc[1] + PATCH_SIZE])

# compute some GLCM properties each patch
xs = []
ys = []
for patch in (grass_patches + sky_patches):
   glcm = graycomatrix(patch, distances=[5], angles=[0],
levels=256,
                       symmetric=True, normed=True)
   xs.append(graycoprops(glcm, 'dissimilarity')[0, 0])
   ys.append(graycoprops(glcm, 'correlation')[0, 0])

# create the figure
fig = plt.figure(figsize=(8, 8))

# display original image with locations of patches
ax = fig.add_subplot(3, 2, 1)
ax.imshow(image, cmap=plt.cm.gray,
          vmin=0, vmax=255)
for (y, x) in grass_locations:
   ax.plot(x + PATCH_SIZE / 2, y + PATCH_SIZE / 2, 'gs')
for (y, x) in sky_locations:
   ax.plot(x + PATCH_SIZE / 2, y + PATCH_SIZE / 2, 'bs')
ax.set_xlabel('Original Image')
ax.set_xticks([])
ax.set_yticks([])
ax.axis('image')

# for each patch, plot (dissimilarity, correlation)
ax = fig.add_subplot(3, 2, 2)
ax.plot(xs[:len(grass_patches)], ys[:len(grass_patches)], 'go',
        label='Grass')
ax.plot(xs[len(grass_patches):], ys[len(grass_patches):], 'bo',
```

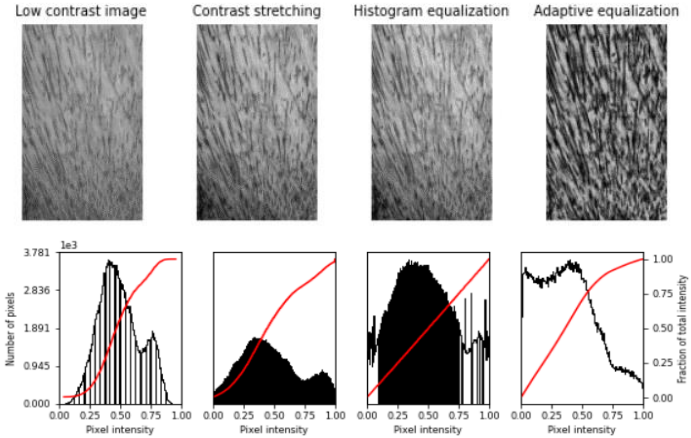
```
        label='Sky')
ax.set_xlabel('GLCM Dissimilarity')
ax.set_ylabel('GLCM Correlation')
ax.legend()

# display the image patches
for i, patch in enumerate(grass_patches):
    ax = fig.add_subplot(3, len(grass_patches),
len(grass_patches)*1 + i + 1)
    ax.imshow(patch, cmap=plt.cm.gray,
              vmin=0, vmax=255)
    ax.set_xlabel('Grass %d' % (i + 1))

for i, patch in enumerate(sky_patches):
    ax = fig.add_subplot(3, len(sky_patches),
len(sky_patches)*2 + i + 1)
    ax.imshow(patch, cmap=plt.cm.gray,
              vmin=0, vmax=255)
    ax.set_xlabel('Sky %d' % (i + 1))


# display the patches and plot
fig.suptitle('Grey level co-occurrence matrix features',
fontsize=14, y=1.05)
plt.tight_layout()
plt.show()
```

D. Persamaan Histogram

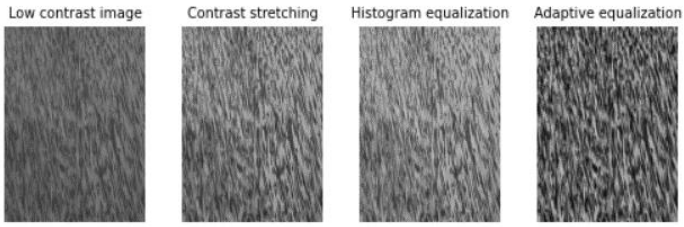
Persamaan histogram digunakan untuk menyempurnakan gambar dengan kontras rendah, menggunakan metode yang disebut pemerataan histogram, hal ini menyebarkan nilai intensitas yang paling sering muncul dalam citra kelapa akan disamakan memiliki fungsi distribusi kumulatif linier.

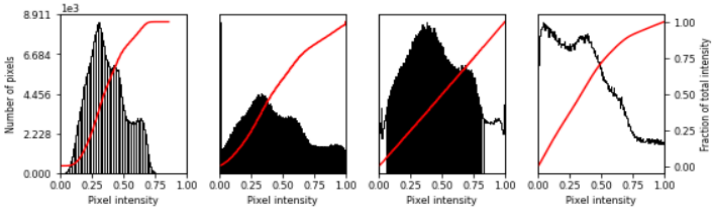


Gambar 9. persamaan histogram (a)

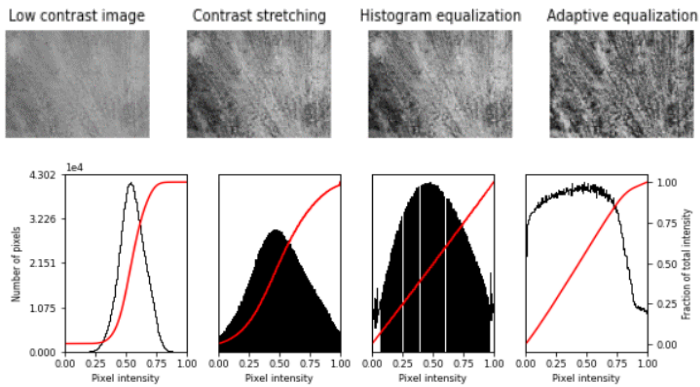


Gambar 10. persamaan histogram (b)





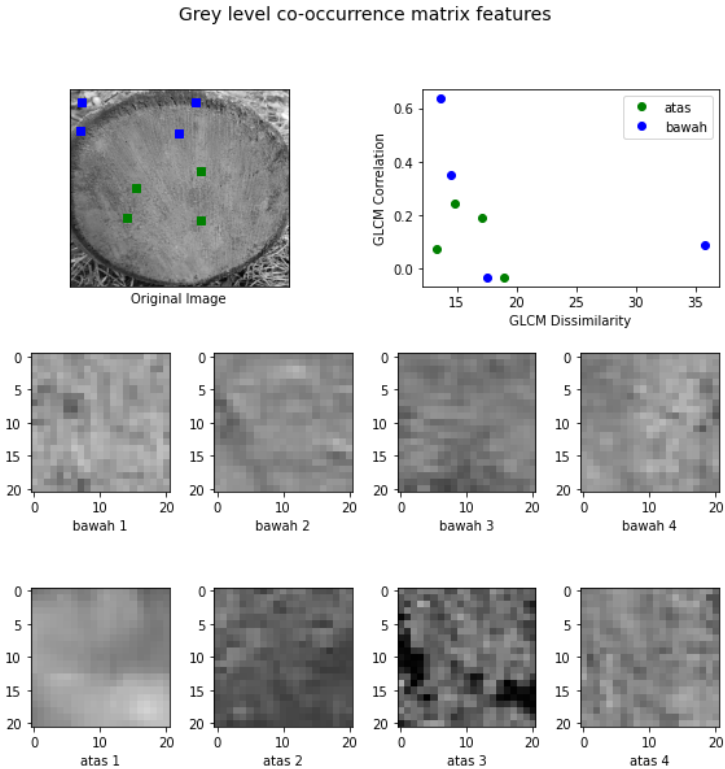
Gambar 11. persamaan histogram (c)



Gambar 12. persamaan histogram (d)

E. Ekstraksi Fitur GLCM

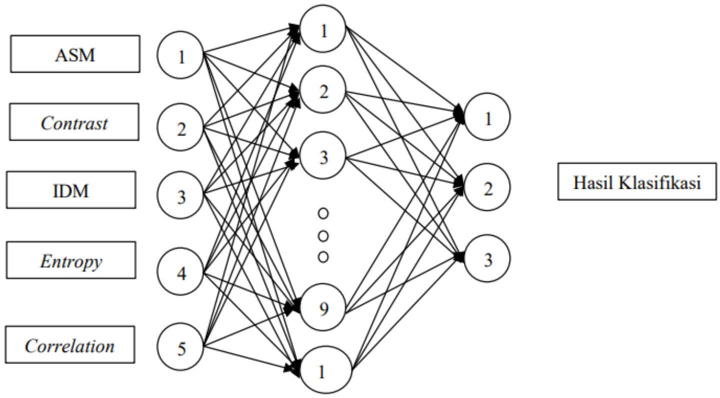
Ekstraksi fitur citra kelapa adalah menggunakan jaringan syaraf tiruan dengan optimasi Genetic Algorithm, penerapan metode akan diawali dengan tahapan pre-prosesing dari gambar kayu dengan menerapakan ekstraksi fitur GLCM untuk fitur dari kayu. Hasil fitur kayu tersebut kemudian dibagi menjadi 2 bagian training dan testing untuk proses pengklasifikasian yang menggunakan metode Jaringan saraf tiruan yang di optimasi dengan algoritma genetik.



Gambar 13. Ekstraksi fitur GLCM

Pada Algoritma genetik, untuk klasifikasi digunakan segmentasi gambar menjadi dua kelas utama:

1. Pemilihan parameter, di mana algoritma genetika digunakan untuk memodifikasi parameter dari metode segmentasi gambar yang ada untuk meningkatkan outputnya.
2. Segmentasi tingkat piksel, di mana algoritma genetika digunakan untuk melakukan pelabelan wilayah. Sebagian besar metode segmentasi gambar memiliki banyak parameter yang perlu dioptimalkan, dan oleh karena itu metode pertama digunakan lebih sering.



Gambar 14 . Arsitektur klasifikasi kayu

1. Angular Second Moment (ASM)

   $ASM = \sum_{i=1}^{L} \sum_{j=1}^{L} (GLCM(i,j))^2$ (1)

2. Contrast

   [illegible] (2)

3. Inverse Difference Moment (IDM)

   [illegible] (3)

4. Entropy

   $Entropy = \sum_{i=1}^{L} \sum_{j=1}^{L} \{GLCM(i,j) \log (GLCM(i,j))\}$ (4)

5. Correlation

   [illegible] (5)

dengan

$\mu i' = \sum_{i=1}^{L} \sum_{j=1}^{L} i * GLCM(i,j)$

$uj' = \sum_{i=1}^{L} \sum_{j=1}^{L} j * GLCM(i,j)$

$\sigma j' = \sum_{i=1}^{L} \sum_{j=1}^{L} GLCM(i,j)(j-\mu j')^2$

$\sigma i' = \sum_{i=1}^{L} \sum_{j=1}^{L} GLCM(i,j)(i-\mu i')^2$

Hasil fitur kayu kemudian dibagi menjadi dua bagian, yaitu pelatihan dan pengujian untuk proses klasifikasi menggunakan metode Jaringan saraf tiruan di optimasi dengan algoritma genetik.

Input dari ekstraksi fitur GLCM menghasilkan output klasifikasi. Klasifikasi gambar yang digunakan adalah gambar kayu. Fitur yang dipilih ini penting untuk mendapatkan performa terbaik berdasarkan metode GLCM. Selain pemilihan fitur, selama pelatihan jaringan, parameter lain sangat signifikan dan perlu bervariasi untuk mendapatkan kinerja terbaik. Parameter yang perlu divariasikan dalam jaringan saraf tiruan pada tahap feedforward dan backforward adalah tingkat pembelajaran yang bervariasi beberapa kali untuk meningkatkan kecepatan di mana jaringan akan belajar.

Hasil ekstraksi fitur warna pada setiap gambar dalam 3 contoh gambar dari 40 total data adalah sebagai berikut.

Id	Fitur	0	45	90	135	Label
04a	asm	2,48E+11	1,86E+11	2,71E+11	1,99E+11	High
	contrast	1,40E+17	2,44E+17	97.586.197.857.773.600	1,96E+17	
	idm	0.142658499130842	0.110306887327680	0.155621702345869	0.116635845716506	
Id	**Fitur**	**0**	**45**	**90**	**135**	**Label**
	entropy	8.709.654.900.190.950	8.975.467.231.102.850	8.576.712.558.184.520	8.889.263.955.162.890	
	correllation	5,98E+11	5,78E+11	6,06E+11	5,87E+11	
56a	asm	1,12E+17	2,18E+17	91.201.738.245.512.200	1,63E+17	Medium
	contrast	0.137232189025026	0.100855255660148	0.151013127024078	0.116189949623670	
	idm	8.721.137.731.165.650	9.053.156.998.709.880	8.652.489.879.567.950	8.907.652.392.841.400	
	entropy	6,03E+11	5,83E+11	6,08E+11	5,94E+11	
	correllation	1,57E+11	2,37E+11	1,82E+11		
75b	asm	4,41E+11	3,65E+11	5,75E+11	3,64E+11	Low
	contrast	63.519.361.238.461.700	98.616.897.267.735.500	41.080.348.882.629.900	98.535.196.644.804.500	
	idm	0.193090788024529	0.158736385981671	0.240168957343864	0.161084794825388	
	entropy	8.211.799.227.788.340	8.406.702.938.022.740	7.976.878.623.731.650	8.403.586.201.616.700	
	correllation	7,23E+11	7,14E+11	7,30E+11	7,14E+11	

Kinerja terbaik diperoleh pada tingkat pembelajaran 0,492. Tingkat akurasi sebelum optimasi diperoleh 67,5% diperoleh dari sistem yang sedang diuji. Hasilnya

ditunjukkan dalam *Confusion Matriks* seperti yang ditunjukkan pada tabel berikut.

Tabel Hasil Confusion Matrix sebelum pengoptimalan

	true low	**true high**	**true medium**	**class precision**
Low pred.	6	0	0	100.00%
High pred.	3	11	4	61.11%
Medium pred.	2	4	10	62.50 %
class recall	54.55%	73.33%	71.43%	

Setelah mengoptimalkan menggunakan GA, menghasilkan akurasi 75%. Hasilnya ditunjukkan dalam *Confusion Matriks* seperti yang ditunjukkan dalam tabel berikut.

Tabel hasil Confusion Matrix setelah pengoptimalan

	true low	**true high**	**true medium**	**class precision**
Low pred.	8	0	2	80%
High pred.	0	12	2	86%
Medium pred.	2	2	10	71%
class recall	80%	86%	71%	

BAB 5
KESIMPULAN

Dengan menggunakan metode segmentasi citra dapat mengidentifikasi tekstur kayu kelapa. Identifikasi kayu kelapa berguna untuk meningkatkan nilai produksi barang olahan yang menggunakan kayu kelapa. Dengan mengetahui tingkat kepadatan kayu menggunakan metode ini dapat menentukan kualitas kayu kelapa. Kualitas kayu kelapa dengan kepadatan tinggi dan medium dapat memberikan nilai informasi bagi pengolah kayu kelapa.

Input dari ekstraksi fitur GLCM menghasilkan output klasifikasi yang kemudian di ujicoba dengan algoritma neural network dan di optimasi dengan algoritma genetic. Berdasarkan hasil percobaan dan pengujian menggunakan algoritma pengolahan citra digital dapat disimpulkan bahwa GA dapat meningkatkan akurasi akurasi.tingkat setelah optimasi menggunakan GA meningkat dari 67,5% menjadi 75%. Selain itu, berdasarkan metode GLCM, fitur seleksi menjadi penting untuk mendapatkan kinerja terbaik. Kinerja terbaik diperoleh pada tingkat pembelajaran 0,492.

REFERENSI

Aravinda, H. L., & Sudhamani, M. V. (2017). Simple Linear Iterative Clustering Based Tumor Segmentation in Liver Region of Abdominal CT-scan. In *Proceedings - 2017 International Conference on Recent Advances in Electronics and Communication Technology, ICRAECT 2017* (pp. 216–222). https://doi.org/10.1109/ICRAECT.2017.18

Azeroual, A., & Afdel, K. (2017). Fast Image Edge Detection based on Faber Schauder Wavelet and Otsu Threshold. *Heliyon*, *July*, 1–19. https://doi.org/10.1016/j.heliyon.2017.e00485

Dimattina, C. (2022). Luminance texture boundaries and luminance step boundaries are segmented using different mechanisms. *Vision Research*, *190*, 107968. https://doi.org/10.1016/j.visres.2021.107968

Houssein, E. H., Helmy, B. E., Oliva, D., Elngar, A. A., & Shaban, H. (2020). A novel Black Widow Optimization algorithm for multilevel thresholding image segmentation. *Expert Systems With Applications*, 114159. https://doi.org/10.1016/j.eswa.2020.114159

Kumar, S., Deep, K., Mirjalili, S., & Thapliyal, S. (2021). Opposition-based Laplacian Equilibrium Optimizer with application in Image Segmentation using Multilevel Thresholding. *Expert Systems With Applications*, *174*(February), 114766. https://doi.org/10.1016/j.eswa.2021.114766

Lei, B., & Fan, J. (2019). Image thresholding segmentation method based on minimum square rough entropy. *Applied Soft Computing Journal*, *84*, 105687. https://doi.org/10.1016/j.asoc.2019.105687

Level, G., Pramunendar, R. A., Supriyanto, C., Novianto, D. H., & Yuwono, I. N. (2013). *Classification Method of Coconut Wood Quality*. *November*, 25–27.

Liu, H., & Singh, J. (2018). Original papers A multispectral machine vision system for invertebrate detection on green leaves. *Computers and Electronics in Agriculture*, *150*(January), 279–288. https://doi.org/10.1016/j.compag.2018.05.002

Marleny, F., & Mambang. (2019). Optimasi Genetic Algorithm Dengan Jaringan Syaraf Tiruan Untuk Klasifikasi Citra. *Jurnal Teknologi Informasi Universitas Lambung Mangkurat (JTIULM)*, *4*(1), 1–6.

Muhammad, K., Ahmad, J., & Baik, S. W. (2018). Early fire detection using convolutional neural networks during surveillance for effective disaster management. *Neurocomputing*, *288*, 1–29. https://doi.org/10.1016/j.neucom.2017.04.083

Riana, D., Rahayu, S., & Hasan, M. (2021). Heliyon Comparison of segmentation and identification of swietenia mahagoni wood defects with augmentation images. *Heliyon*, *7*(January), e07417. https://doi.org/10.1016/j.heliyon.2021.e07417

Shahabi, H., Shirzadi, A., Ronoud, S., Asadi, S., Pham, B. T., Mansouripour, F., Geertsema, M., Clague, J. J., & Bui, D. T. (2021). Flash flood susceptibility mapping using a novel deep learning model based on deep belief network, back propagation and genetic algorithm. *Geoscience Frontiers*, *12*(3), 101100. https://doi.org/10.1016/j.gsf.2020.10.007

Srivaro, S., Lim, H., Li, M., Jantawee, S., & Tomad, J. (2021). Effect of compression ratio and original wood density on pressing characteristics and physical and mechanical properties of thermally compressed coconut wood. *Construction and Building Materials*, *299*(May), 124272. https://doi.org/10.1016/j.conbuildmat.2021.124272

Sun, Y., Gao, J., Wang, K., Shen, Z., & Chen, L. (2018). *Utilization of Machine Vision to Monitor the Dynamic Responses of Rice Leaf Morphology and Colour to Nitrogen , Phosphorus , and Potassium Deficiencies*. *2018*, 16–18.

Tamal, M. (2020). Heliyon Intensity threshold based solid tumour segmentation method for Positron Emission Tomography (PET) images ; A review. *Heliyon*, *6*(May), e05267. https://doi.org/10.1016/j.heliyon.2020.e05267

Tian, H., Wang, T., Liu, Y., Qiao, X., & Li, Y. (2020). Computer vision technology in agricultural automation — A review. *Information Processing in Agriculture*, *7*(1), 1–19. https://doi.org/10.1016/j.inpa.2019.09.006

Wang, Y., Zhang, X., Ma, G., Du, X., Shaheen, N., & Mao, H. (2021). *Recognition of weeds at asparagus fields using multi-feature fusion and backpropagation neural network*. *14*(4), 190–198. https://doi.org/10.25165/j.ijabe.20211404.6135

Wu, B., Zhou, J., Ji, X., Yin, Y., Shen, X., Zhou, J., Ji, X., Yin, Y., & Shen, X. (2020). An ameliorated teaching-learning-based optimization algorithm based study of image segmentation for multilevel thresholding using Kapur ' s entropy and Otsu ' s between class variance. *Information Sciences*. https://doi.org/10.1016/j.ins.2020.05.033

Wu, T., Shao, J., Gu, X., Ng, M. K., & Zeng, T. (2021). Two-stage image segmentation based on nonconvex approximation and thresholding. *Applied Mathematics and Computation*, *403*, 1–18. https://doi.org/10.1016/j.amc.2021.126168

Yang, Y., Zhao, X., Huang, M., Wang, X., & Zhu, Q. (2021). Multispectral image based germination detection of potato by using supervised multiple threshold segmentation model and Canny edge detector. *Computers and Electronics in Agriculture*, *182*(February), 106041. https://doi.org/10.1016/j.compag.2021.106041

Zhao, S., Wang, P., Asghar, A., Chen, H., Turabieh, H., Mafarja, M., & Li, C. (2021). Multilevel threshold image segmentation with diffusion association slime mould algorithm and Renyi ' s entropy for chronic obstructive pulmonary disease. *Computers in Biology and Medicine*, *134*(May), 104427. https://doi.org/10.1016/j.compbiomed.2021.104427

PROFIL PENULIS

Finki Dona Marleny kelahiran Kijang (*kepulauan Riau*). Ketika Masih Kecil ia tinggal di Muara Enim Sumatera selatan dan melanjutkan sekolah dasar di Talawi, Sawahlunto Sumatera Barat hingga kelas 3 SD. Kemudian pindah ke kota kelahirannya di Kijang Pulau Bintan Kepulauan Riau dan menamatkan Pendidikan SD serta melanjutkan sekolah ke tingkat SMP hingga kelas 1. Pada tahun 2002 ia pindah kembali ke Kabupaten Balangan Kalimantan Selatan dan pada tahun 2006 melanjutkan studi di STMIK INDONESIA Banjarmasin Jurusan Sistem Informasi. Kemudian Ia mendapatkan beasiswa di salah satu Yayasan dan melanjutkan studi S2 di jurusan Teknik Informatika UDINUS (Universitas Dian Nuswantoro Semarang lulus pada tahun 2012. Sampai sekarang masih aktif sebagai *Content Creator, blogger* dan Dosen di salah satu Universitas Swasta di kota Banjarmasin.

Hartini lahir di Karanganyar (Jawa Tengah) pada tanggal 02 April 1985. Pada Tahun 1997 lulus Sekolah Dasar Negeri 2 Nglebak, dilanjutkan Sekolah Lanjutan Tingkat Pertama lulus pada Tahun 2000 dan pada tahun 2003 lulus Sekolah Menegah Atas di SMA Warga Surakarta. Tertarik dan menekuni pembelajaran di bindang Teknologi Informasi maka saya melanjutkan kuliah strata-1 di STMIK Adi Ungggul Bhirawa (AUB) Surakarta dan pada 2012 saya menyelesaikan strata-2 di Magister Sistem Informasi Universitas Diponegoro Semarang. Untuk mengembangkan ilmu yang didapat maka saya mengabdikan diri sebagai dosen informatika pada Program Studi Manajemen Informatika Politeknik Muara Teweh Kabupaten Barito Utara Kalimantan Tengah.

Ayu Ahadi Ningrum lahir di Martapura (Kalimantan Selatan) pada yanggal 03 Pebruari 1990. Pada Tahun 2002 lulus sekolah dasar dari SDN Jawa 2 Martapura, melanjutkan Sekolah Menengah Pertama di SMPN 1 Martapura lulus ditahun 2005 dan tahun 2008 lulus pada SMKN 1 Martapura jurusan Multimedia. Pada tahun 2012 penulis lulus dari Sekolah Tinggi Ilmu Ekonomi Pancasetia jurusan Akutansi, lalu 2013 lulus dari program D-III jurusan Teknologi Informasi di Politeknik Negeri Banjarmasin. Sempat bekerja selama 6 tahun pada sektor Mining pada bagian Manajemen Informasi Sistem sebagai System Analyst, tahun 2018 kembali lulus program D-IV di Politeknik Elektonika Negeri Surabaya dan melanjutkan strata-2 terapan di Pascasarjana Terapan Politeknik Eletronika Surabaya. Untuk menerapkan serta mengembangkan ilmu yang sudah didapat, maka penulis melanjutkan karir menjadi dosen informatika pada Program Studi Informatika Fakultas Teknikn Universitas Mhammadyah Banjarmasin.

Ihdalhubbi Maulida lahir di Banjarmasin (Kalimantan Selatan) pada tanggal 30 September 1990. Pada Tahun 2002 lulus sekolah dari Madrasah Diniyah Islamiyah Muhammadiyah 1-2 Sungai Kindaung Banjarmasin, dilanjutkan Sekolah Menengah Pertama di SMP Negeri 24 Banjarmasin di tahun 2002 dan tahun 2008 lulus pada Sekolah Menengah Atas di SMA Negeri 2 Banjarmasin. Minat kepada pembelajaran tentang teknologi maka saya melanjutkan kuliah strata-1 di STMIK Indonesia Banjarmasin pada tahun 2008 dan pada tahun 2014 saya menyelesaikan strata-2 di Universitas Dian Nuswantoro Semarang. Untuk mengembangkan ilmu yang didapat maka saya

mengabdikan diri saya sebagai dosen informatika pada Program Studi Informatika Fakultas Teknik Universitas Muhammadiyah Banjarmasin.

Rudy Ansari merupakan dosen yang mendapatkan gelar insinyur teknik di Universitas Lambung Mangkurat. Saat ini aktif mengajar di program studi informatika fakultas teknik Universitas Muhammadiyah Banjarmasin. Rudy Ansari kelahiran Buntok memiliki gelar sarjana teknik informatika di kota Malang yaitu di Sekolah Tinggi Informatika & Komputer Indonesia dan melanjutkan studi pasca sarjana di universitas Dian Nuswantoro Semarang. Selain aktif di bidang insinyur Teknik dan informatika, Bidang riset yang ditekuni adalah machine learning, data science, dan software engineering. Dapat dihubungi melalui kontak melalui email: rudy@umbjm.ac.id

Mambang, M.Kom merupakan Dosen Universitas Sari Mulia sejak 2009 sampai dengan sekarang. Selain sebagai akademisi, penulis juga memiliki CV. Mediatikom Banua Group yang bergerak dalam bidang pelatihan IT di kota Banjarmasin dan sekitarnya. Penulis juga menjabat sebagai Ketua Program Studi Teknologi Informasi 2019-2023. Selain itu juga menjabat sebagai ketua IndoCEISS 2021-2025. Penulis telah menerbitkan beberapa karya buku diantaranya: Buku Ajar Teknologi Komunikasi Internet (IoT) tahun 2021, Buku Ajar Konsep Dasar Teknologi Informasi (2021), Tantangan dan peluang generasi muda di era 4.0 (2020), 35 pekerjaan bidang IT di era 4.0 (2020). Penulis juga aktif dalam menerbitkan opini pada media massa online tingkat Wilayah dan juga Nasional serta membuat

publikasi artikel pada jurnal Nasional dan International. Penulis dapat dihubungi melalui email: mmbg1283@gmail.com / mambang@unism.ac.id.