DAFTAR PUSTAKA

Amirullah, D. (2018). Sistem Pencarian Semantik Impresi dengan Mekanisme Pembobotan Kombinasi Fitur Warna dan Fitur Bentuk. *INOVTEK Polbeng - Seri Informatika*, *3*(1), 41, https://doi.org/10,35314/isi.v3i1,332

Annas T.S., T. (2019). *Perbandingan Model Warna RGB, HSL dan HSV Sebagai Fitur dalam Prediksi Cuaca pada Citra Langit menggunakan K-Means.*

Badan Standarisasi Nasional. (2008). *BIJI KAKAO*.

C. Gonzalez, R., & E. Woods, R. (2008). *Digital Image Processing Third Edition*. Pearson Education.

Chih-Wei Hsu, C.-C. C. and C.-J. L. (2003). *A Practical Guide to Support Vector Classification*.

DIANSARI, A. Z. (2015). *KARAKTERISTIK FISIK, KIMIA DAN MIKROBIOLOGIS BIJI KAKAO KERING PRODUKSI PTPN XII KEBUN KALIKEMPIT-BANYUWANGI*.

DITJENBUN. (2019). *Kementerian Pertanian Direktorat Jenderal Perkebunan » Cokelatku Budayaku Indonesiaku : TUMBUHKAN BUDAYA KORPORASI PEKEBUN KAKAO*. https://ditjenbun.pertanian.go.id/cokelatku-budayaku-indonesiaku-tumbuhkan-budaya-korporasi-pekebun-kakao/

Fetia Wardhiani, W. (n.d.). *PERAN POLITIK PERTANIAN DALAM PEMBANGUNAN PERTANIAN MENGHADAPI ERA REVOLUSI INDUSTRI 4.0 DI SEKTOR PERTANIAN*.

Hapsari, O. A. (2023). *Berita BISIP - Kakao Indonesia: Produksi, Tantangan dan Peluang*. https://bisip.bsip.pertanian.go.id/berita/kakao-indonesia-produksi-tantangan-dan-peluang

Henry A. (2014). *Pengolahan Citra Digital: Konsep Dasar Representasi Citra*. https://slideplayer.info/slide/1966651/

Hidayat, A., Makhsun, D., & M.Si. (2018). Analisis Citra Daun Berdasarkan Fitur Local Binary Pattern dan Fitur Canny Edge Detection Menggunakan Metode Klasifikasi K-Nearest Neighbor (K-NN). *STMIK Eresha*. https://vbook.pub/documents/analisis-citra-daun-berdasarkan-fitur-local-

binary-pattern-dan-fitur-canny-edge-detection-menggunakan-metode-klasifikasi-k-nearest-neighbor-k-nn-ko75r9qkm3w3

Hilyatunnisa, I. (2013). *Pengaruh Adenin (6 - Amino Purine) Terhadap Keberhasilan Embriogenesis Somatik Bunga Kakao (Theobroma cacao L.)*. Universitas Muhammadiyah Purwokerto.

Indrabulan, T. (2016). *Perbandingan Metode Deteksi Objek Berbasis Video untuk Survei Arus Lalu Lintas*. Universitas Hasanuddin.

I Nyoman Wata. (2018). *PENINGKATAN MUTU BIJI KAKAO UNTUK MENOPANG HARGA JUAL KAKAO*.

Jonathan Sarwono. (2018). *Metode Penelitian Kuantitatif dan Kualitatif* (2nd ed.). Suluh Media.

KEMENTAN. (2019). *Mentan: Alsintan mendukung revolusi industri 4.0,* Https://Pertanian.Go.Id/Home/?Show=news&act=view&id=3736.

K. Jain, A. (1989). *Fundamental of Digital Image Processing*. Prentice-Hall.

Kusuma, S. F., Pawening, R. E., & Dijaya, R. (2017). Otomatisasi klasifikasi kematangan buah mengkudu berdasarkan warna dan tekstur. *Register: Jurnal Ilmiah Teknologi Sistem Informasi*, *3*(1), 17–23. https://doi.org/10,26594/register.v3i1,576

Li, G., You, J., & Liu, X. (2015). Support Vector Machine (SVM) based prestack AVO inversion and its applications. *Journal of Applied Geophysics*, *120*, 60–68. https://doi.org/https://doi.org/10,1016/j.jappgeo.2015.06.009

Prabowo, D. A., & Abdullah, D. (2018). Deteksi dan Perhitungan Objek Berdasarkan Warna Menggunakan Color Object Tracking. *Pseudocode*, *5*(2), 85–91, https://doi.org/10,33369/pseudocode.5.2.85-91

Praseptiyana, W. I., Widodo, A. W., & Rahman, M. A. (2019). Pemanfaatan Ciri Gray Level Co-occurrence Matrix (GLCM) Untuk Deteksi Melasma Pada Citra Wajah. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *3*(11), 10402–10409.

Prasetyo, E. (2012). *Data mining: konsep dan aplikasi menggunakan MATLAB* (1st ed.). CV Andi Offset.

Prawira, A. B., Jayanta, & Widiastiwi, Y. (2021). Penerapan Metode Gray Level Co-Occurance Matrix dan Algoritma Support Vector Machine Pada

Klasifikasi Tanaman Bidara Berdasarkan Tekstur Daun. *Seminar Nasional Mahasiswa Ilmu Komputer Dan Aplikasinya (SENAMIKA)*, 569.

PT. Freyabadi. (2022). *Proses Pengolahan Coklat dari Biji Kakao*. Https://Www.Freyabadi.Com/Id/Blog/Proses-Pengolahan-Coklat-Dari-Biji-Kakao.

Purnama, A. L. I. (2021). *Proses Pengolahan dan Analisis Mutu Biji Kakao Edel (Theobroma cacao L.) Di PT. Perkebunan Nusantara XII (PERSERO) Kebun Bantaran Afdeling Penataran Blitar*.

Putri, R. Z. (2023). *Klasifikasi Tanaman Obat Menggunakan Multiclass Support Vector Machine Berbasis Android* (Issue D121 17 1001).

Rachmat, M. P. (2018). *Sistem Pengenalan Wajah Untuk Monitoring Pegawai*.

Radifan, M. R. (2022). *Deteksi Dan Hitung Buah Cokelat Matang dan Non-Matang Pada Pohonnya Secara Realtime* [Universitas Hasanuddin]. http://repository.unhas.ac.id/id/eprint/32409/

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. https://doi.org/10,1109/TPAMI.2016.2577031

Ruang Warna Hue Saturation Value (HSV) serta Proses Konversinya. (n.d.). Retrieved October 23, 2024, from https://www.kitainformatika.com/2015/01/ruang-warna-hue-saturation-value-hsv.html

Sarwono, J. (2018). *Metode penelitian kuantitatif dan kualitatif*. Suluh Media.

Schober, P., Boer, C., & Schwarte, L. (2018a). Correlation Coefficients: Appropriate Use and Interpretation. *Anesthesia & Analgesia*, *126*, 1, https://doi.org/10,1213/ANE.0000000000002864

Schober, P., Boer, C., & Schwarte, L. A. (2018b). Correlation Coefficients: Appropriate Use and Interpretation. *Anesthesia & Analgesia*, *126*(5), 1763–1768. https://doi.org/10,1213/ANE.0000000000002864

Schölkopf, B., & Smola, A. J. (2001). *Learning with Kernels*. The MIT Press. https://doi.org/10,7551/mitpress/4175.001,0001

Singh, P., Singh, N., Singh, K. K., & Singh, A. (2021). Chapter 5 - Diagnosing of disease using machine learning. In K. K. Singh, M. Elhoseny, A. Singh, & A. A. Elngar (Eds.), *Machine Learning and the Internet of Medical Things in Healthcare* (pp. 89–111). Academic Press. https://doi.org/https://doi.org/10,1016/B978-0-12-821229-5.00003-3

Sulistiyanti, S., Setyawan, F. A., & Komarudin, M. (2016). *PENGOLAHAN CITRA DASAR DAN CONTOH PENERAPANNYA*.

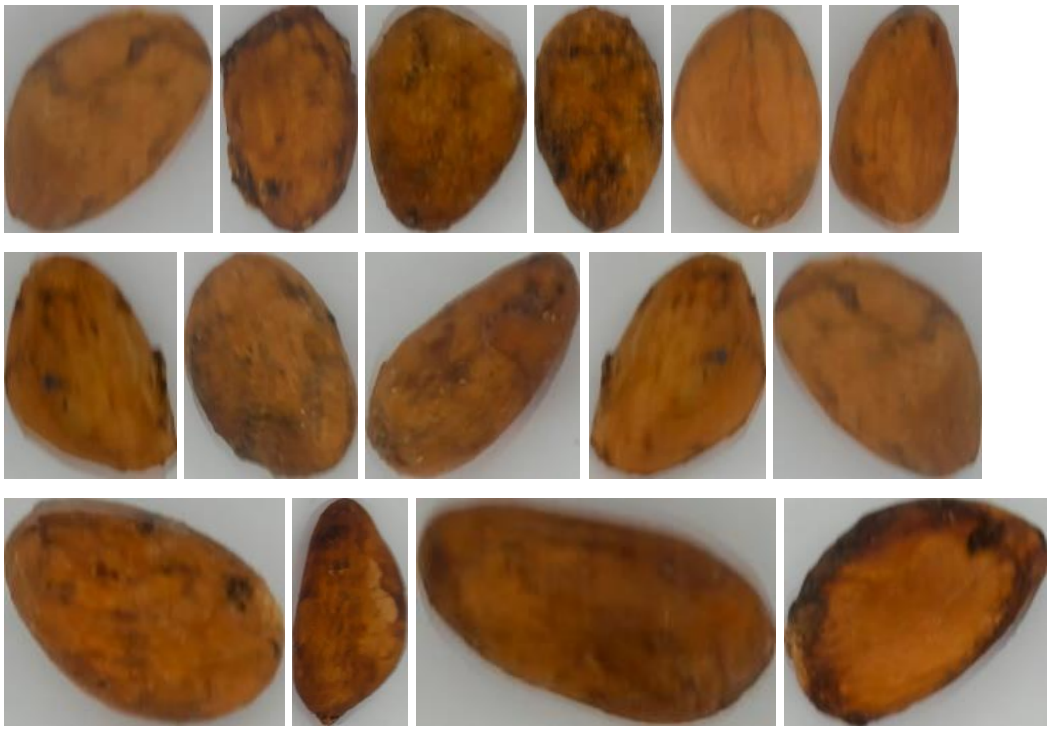
Sumarlin, S. (2015). Implementasi Algoritma K-Nearest Neighbor Sebagai Pendukung Keputusan Klasifikasi Penerima Beasiswa PPA dan BBM. *JURNAL SISTEM INFORMASI BISNIS*, *5*(1). https://doi.org/10,21456/vol5iss1pp52-62

Suzuki, S., & be, K. (1985). Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, *30*(1), 32–46. https://doi.org/10,1016/0734-189X(85)90016-7

Yanu F., M., Yuwono, B., & Boedi P., D. (2022). *Dasar Pengolahan Citra Digital Edisi 2022*. Lembaga Penelitian dan Pengabdian kepada Masyarakat.

Zinzendoff Okwonu, F., Laro Asaju, B., & Irimisose Arunaye, F. (2020). Breakdown Analysis of Pearson Correlation Coefficient and Robust Correlation Methods. *IOP Conference Series: Materials Science and Engineering*, *917*(1), 012065. https://doi.org/10,1088/1757-899X/917/1/012065

Lampiran 1 Contoh *Frame* Biji Kakao Pada Proses *Training*



Frame Biji Kakao Baik

Frame Biji Kakao Buruk

Lampiran 2 Nilai Ekstraksi Fitur

Scan QR File Nilai Ekstraksi Fitur

Lampiran 3 Gambar Hasil *Confussion Matrix* Untuk Setiap Skenario

Scan QR Gambar *Confussion Matrix*

Lampiran 4 Video Pengambilan Data dan Hasil Testing

Scan QR Video Testing

Lampiran 5 *Source Code*

1. Ekstraksi Video Ke *Frame*

```
import cv2

import os

# Path lengkap ke file video

video_file = r'F:\PENELITIAN\SKRIPSI\DATA\6. SKENARIO TRAINING \SKENARIO PENGAMBILAN DATA\5cm_80rpm.mp4'

# Nama folder untuk menyimpan frame

output_folder = r'F:\PENELITIAN\SKRIPSI\6. SKENARIO TRAINING \SKENARIO PENGAMBILAN DATA \
FIX_skenario_5cm_80rpm\'

# Pastikan folder output sudah ada atau buat jika belum ada

os.makedirs(output_folder, exist_ok=True)

# Buka video

video_capture = cv2.VideoCapture(video_file)

# Pastikan video terbuka dengan sukses

if not video_capture.isOpened():

   print("Gagal membuka video")

   exit()

# Loop untuk mengekstrak frame

frame_count = 0

while True:

   ret, frame = video_capture.read()

   # Keluar dari loop jika tidak ada frame lagi

   if not ret:

      break

   # Simpan frame dalam folder

   frame_filename = os.path.join(output_folder, f'frame_{frame_count:04d}.jpg')

   cv2.imwrite(frame_filename, frame)

   frame_count += 1

# Tutup video dan jendela tampilan jika ada

video_capture.release()

cv2.destroyAllWindows()

print(f"{frame_count} frame telah diekstrak dan disimpan dalam folder '{output_folder}'.")
```

2. *Source Code Preprocessing*

```
import cv2
import numpy as np
import os

def detect_and_save_contours(input_folder, output_base_folder, h_range, s_range, v_range, min_contour_area=500):
    # Membuat folder output untuk mask, contour, bounding box, dan crop images
    output_folder_contour = os.path.join(output_base_folder, 'contour')
    output_folder_bbox = os.path.join(output_base_folder, 'bounding_box')
    output_folder_crop_flip = os.path.join(output_base_folder, 'crop+flip')
    # output_folder_flip = os.path.join(output_base_folder, 'flip')  # Folder untuk hasil flip

    # Buat folder jika belum ada
    for folder in [output_folder_contour, output_folder_bbox, output_folder_crop_flip]:
        if not os.path.exists(folder):
            os.makedirs(folder)

    # Definisi batas-batas untuk mask HSV
    lower_bound = np.array([h_range[0], s_range[0], v_range[0]])
    upper_bound = np.array([h_range[1], s_range[1], v_range[1]])

    resize_dim=(960, 540)
    for filename in os.listdir(input_folder):
        if filename.endswith(('.jpg', '.png', '.jpeg', '.bmp')):
            input_path = os.path.join(input_folder, filename)
            image_bgr = cv2.imread(input_path)
            if image_bgr is None:
                print(f"Cannot read image from {input_path}")
                continue

            # Resize image
            image_bgr = cv2.resize(image_bgr, resize_dim)

            # Convert BGR to RGB then to HSV
            image_rgb = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2RGB)
            image_hsv = cv2.cvtColor(image_rgb, cv2.COLOR_RGB2HSV)

            # Create a mask based on the defined HSV range
            mask = cv2.inRange(image_hsv, lower_bound, upper_bound)
            masked_image = cv2.bitwise_and(image_rgb, image_rgb, mask=mask)

            # Find contours from the mask
            contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

            # Filter out small contours
            filtered_contours = [contour for contour in contours if cv2.contourArea(contour) > min_contour_area]

            image_contour = image_rgb.copy()
```

```
        cv2.drawContours(image_contour, filtered_contours, -1, (0, 255, 0), 2)

        # Save the contour image
        output_path_contour = os.path.join(output_folder_contour, filename)
        cv2.imwrite(output_path_contour, cv2.cvtColor(image_contour, cv2.COLOR_RGB2BGR))

        image_bbox = image_rgb.copy()
        for contour in filtered_contours:
            x, y, w, h = cv2.boundingRect(contour)
            cv2.rectangle(image_bbox, (x, y), (x+w, y+h), (255, 0, 0), 2)
            crop_img = image_rgb[y:y+h, x:x+w]

            # Save the bounding box image and cropped image
            output_path_bbox = os.path.join(output_folder_bbox, filename)
            output_path_crop = os.path.join(output_folder_crop_flip, filename)
            cv2.imwrite(output_path_bbox, cv2.cvtColor(image_bbox, cv2.COLOR_RGB2BGR))
            cv2.imwrite(output_path_crop, cv2.cvtColor(crop_img, cv2.COLOR_RGB2BGR))

            # Augmentasi dengan flipping horizontal
            flip_img = cv2.flip(crop_img, 1)  # 1 untuk flipping horizontal
            output_path_flip = os.path.join(output_folder_crop_flip, f"flip-{filename}")
            cv2.imwrite(output_path_flip, cv2.cvtColor(flip_img, cv2.COLOR_RGB2BGR))

# HSV range for masking
h_range = (0, 179)
s_range = (56, 255)  # s_min bad seeds = 25, s_min good seeds = 56
v_range = (0, 255)

input_folder = r'F:\PENELITIAN\SKRIPSI\DATA\6. SKENARIO TRAINING\5. SKENARIO PENGAMBILAN DATA\FIX_skenario_5cm_80rpm\Biji Baik'
output_base_folder = r'F:\PENELITIAN\SKRIPSI\DATA\6. SKENARIO TRAINING\5. SKENARIO PENGAMBILAN DATA\FIX_INPUT_5cm_80rpm\Biji Baik'

# Run the conversion and contour detection function
detect_and_save_contours(input_folder, output_base_folder, h_range, s_range, v_range)
```

3. *Source Code* Ekstraksi Fitur

```
import cv2
import numpy as np
import pandas as pd
from skimage.feature import greycomatrix, greycoprops
from scipy import stats
import os


def create_hsv_glcm_dataset(base_folder):
    # Definisi kolom untuk dataset
    glcm_feature = ['contrast', 'homogeneity', 'ASM', 'energy', 'correlation', 'dissimilarity']
    angle = ['0', '45', '90', '135']

    # Membuat kolom dataset untuk fitur HSV dan GLCM
    names = ['class', 'hue', 'saturation', 'value']
    for i in glcm_feature:
        for j in angle:
            names.append(i + ' ' + j)

    # Membuat DataFrame berdasarkan nama kolom yang dibuat
    df = pd.DataFrame(columns=names)

    # Menyiapkan path untuk folder Biji Baik dan Biji Buruk
    class_folders = {'Biji Buruk': 0, 'Biji Baik': 1}

    for folder_name, label in class_folders.items():
        folder_path = os.path.join(base_folder, folder_name)
        for filename in os.listdir(folder_path):
            image_path = os.path.join(folder_path, filename)

            # Membaca gambar dan mengubahnya menjadi format HSV
            img = cv2.imread(image_path)
            if img is None:
                print(f"Image at path {image_path} could not be loaded.")
                continue

            hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
            # Ekstraksi fitur HSV
            height, width = hsv.shape[:2]
            H = hsv[:, :, 0]  # Hue channel
            S = hsv[:, :, 1]  # Saturation channel
            V = hsv[:, :, 2]  # Value channel

            hue = H.flatten()  # Mengubah H menjadi vektor 1D
            mode_h = stats.mode(hue)
            mode_hue = int(mode_h.mode) if int(mode_h.mode) != 0 else np.mean(H)
            mean_s = np.mean(S)
```

```
            mean_v = np.mean(V)

            # Konversi gambar ke grayscale untuk ekstraksi GLCM
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

            # Ekstraksi fitur GLCM
            distance = [5]
            angles = [0, np.pi/4, np.pi/2, 3*np.pi/4]
            glcm = greycomatrix(gray, distances=distance, angles=angles, levels=256, symmetric=True, normed=True)
            glcm_props = []
            for name in glcm_feature:
                for angle in range(len(angles)):
                    glcm_props.append(greycoprops(glcm, prop=name)[0, angle])

            # Menyusun data ke dalam satu vektor untuk satu gambar
            vector = [label] + [mode_hue, mean_s, mean_v] + glcm_props

            # Menambahkan vektor ke DataFrame
            df_temp = pd.DataFrame([vector], columns=names)

            if not df_temp.isnull().all().all():  # Memastikan df_temp tidak kosong atau seluruhnya NA
                df = pd.concat([df, df_temp], ignore_index=True)


    return df

# Path utama yang berisi folder 'Biji Baik' dan 'Biji Buruk'
base_folder = 'FIX_INPUT_5cm_80rpm'

df = create_hsv_glcm_dataset(base_folder)
pd.DataFrame(df)


# Simpan DataFrame ke dalam file CSV
output_csv_path = 'FIX_data_skenario_5cm_80rpm.csv'
df.to_csv(output_csv_path, index=False)
```

4. *Source Code Training* SVM

```
#Library
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn import svm
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_auc_score, roc_curve, auc
import seaborn as sns
import matplotlib.pyplot as plt
import time

klasifikasiSVM_start = time.time()

# Memuat data train pada dataframe
df_train = pd.read_csv('FIX_data_skenario_5cm_80rpm.csv')

# Menyimpan fitur atribut ke dalam variabel X dan class label pada y
X = df_train.drop(labels=['class'], axis=1)
y = df_train['class']

# Melakukan pembagian data dengan train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)

# Definisi parameter untuk GridSearchCV
parameters = {
   'kernel': ['rbf'],
   'gamma': [1e-3, 0.01, 0.1, 0.5, 1, 10],
   'C': [0.1, 1, 10, 100, 1000]
}

# Menggunakan GridSearch dengan memanggil class SVC
model_param = GridSearchCV(svm.SVC(probability=True), parameters, cv=5, return_train_score=True)

# Melakukan training pada objek dan label
model_param.fit(X_train, y_train)

# Menampilkan hasil dari model SVM berdasarkan Hyper Parameter Tuning
means = model_param.cv_results_['mean_test_score']
```

```
stds = model_param.cv_results_['std_test_score']

print("Hasil Hyperparameter Tuning:")
for mean, std, params in zip(means, stds, model_param.cv_results_['params']):
   print('%0.3f (+/-%0.03f) untuk %r' % (mean, std * 2, params))

# Menampilkan hasil terbaik dari model SVM berdasarkan Hyperparameter Tuning
print("\nHasil GridSearchCV:")
print("Best parameters found: ", model_param.best_params_)
print("Best score: ", model_param.best_score_)

# Visualisasi hasil GridSearchCV dengan heatmap
cv_results = pd.DataFrame(model_param.cv_results_)
cv_results['param_C'] = cv_results['param_C'].astype(float)
cv_results['param_gamma'] = cv_results['param_gamma'].astype(float)

heatmap_data = cv_results.pivot_table(index='param_gamma', columns='param_C', values='mean_test_score')

plt.figure(figsize=(8, 6))
sns.heatmap(heatmap_data, annot=True, fmt=".3f", cmap="viridis", cbar_kws={'label': 'Mean Test Score'})
plt.title('Grid Search CV Results', fontsize=16)
plt.xlabel('Parameter C', fontsize=12)
plt.ylabel('Parameter gamma', fontsize=12)
plt.show()

# Evaluasi model terbaik pada data uji
y_pred = model_param.predict(X_test)
print("\nAkurasi pada data uji: ", accuracy_score(y_test, y_pred))

# Menampilkan laporan klasifikasi
print("\nLaporan Klasifikasi:\n", classification_report(y_test, y_pred))

# Membuat confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Menentukan jumlah kelas yang ada dalam data
num_classes = len(np.unique(y))
```

```
# Membuat DataFrame untuk confusion matrix dengan indeks dan kolom yang sesuai dengan jumlah kelas aktual
conf_matrix_df = pd.DataFrame(conf_matrix, index=[str(i) for i in range(num_classes)], columns=[str(i) for i in range(num_classes)])

# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_df, annot=True, annot_kws={'size': 12}, fmt='d', cmap='Blues')
plt.title('Confusion Matrix untuk Model SVM dengan GridSearchCV\n', fontsize=18)
plt.ylabel('True label', fontsize=14)
plt.xlabel('Predicted label', fontsize=14)
plt.show()

# Menghitung skor ROC-AUC dan kurva ROC menggunakan decision_function
y_score = model_param.decision_function(X_test)
fpr, tpr, _ = roc_curve(y_test, y_score)
roc_auc = auc(fpr, tpr)

# Membuat plot untuk ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc="lower right")

plt.show()
print('ROC: ', roc_auc)
klasifikasiSVM_end = time.time()
print(f"Waktu Komputasi adalah {klasifikasiSVM_end - klasifikasiSVM_start} detik")
```

5. *Source Code Testing*

```
import cv2
import joblib
import numpy as np
from skimage.feature import greycomatrix, greycoprops
from scipy import stats
import matplotlib.pyplot as plt

# Fungsi untuk mendeteksi dan memotong kontur dengan validasi ROI
def detect_and_crop_contours(frame, h_range, s_range, v_range, min_contour_area=500):
    # Resize frame ke ukuran yang diinginkan hanya untuk keperluan ekstraksi fitur
    resize_dim = (960, 540)
    resized_frame = cv2.resize(frame, resize_dim)

    # Mendefinisikan ROI (Region of Interest) sebagai area tengah dari frame yang telah di-resize
    frame_height, frame_width, _ = resized_frame.shape
    roi_x1, roi_y1 = int(frame_width * 0.25), int(frame_height * 0.25)
    roi_x2, roi_y2 = int(frame_width * 0.75), int(frame_height * 0.75)

    # Konversi BGR ke HSV
    image_hsv = cv2.cvtColor(resized_frame, cv2.COLOR_BGR2HSV)

    # Definisi batas untuk masking pada HSV
    lower_bound = np.array([h_range[0], s_range[0], v_range[0]])
    upper_bound = np.array([h_range[1], s_range[1], v_range[1]])

    # Membuat mask berdasarkan range HSV yang didefinisikan
    mask = cv2.inRange(image_hsv, lower_bound, upper_bound)

    # Menemukan kontur dari mask
    contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    # Filter kontur kecil
    filtered_contours = [contour for contour in contours if cv2.contourArea(contour) > min_contour_area]

    # Jika ada kontur yang terdeteksi, lakukan cropping pada bounding box pertama yang ditemukan
    if filtered_contours:
        x, y, w, h = cv2.boundingRect(filtered_contours[0])

        # Cek apakah bounding box berada di dalam ROI
        if roi_x1 <= x <= roi_x2 and roi_y1 <= y <= roi_y2:
            cropped_img = resized_frame[y:y+h, x:x+w]
            # Gambar bounding box pada gambar asli (bukan di-resize)
            x_original = int(x * (frame.shape[1] / resized_frame.shape[1]))
            y_original = int(y * (frame.shape[0] / resized_frame.shape[0]))
            w_original = int(w * (frame.shape[1] / resized_frame.shape[1]))
            h_original = int(h * (frame.shape[0] / resized_frame.shape[0]))
            image_bgr_with_bbox = frame.copy()
```

```
            cv2.rectangle(image_bgr_with_bbox, (x_original, y_original), (x_original + w_original, y_original + h_original), (255,
255, 0), 2)
            return cropped_img, image_bgr_with_bbox
        else:
            print("Bounding box is not within ROI.")
            return None, frame
    else:
        print("No contours found or contours are too small.")
        return None, frame

# Fungsi untuk ekstraksi fitur dari citra hasil cropping
def extract_hsv_glcm_features(cropped_img):
    if cropped_img is None:
        print("Cropped image is None, skipping feature extraction.")
        return None

    # Mengubah gambar menjadi format HSV
    hsv = cv2.cvtColor(cropped_img, cv2.COLOR_BGR2HSV)

    # Ekstraksi fitur HSV
    height, width = hsv.shape[:2]
    S = hsv[:, :, 1]  # Saturation channel
    V = hsv[:, :, 2]  # Value channel
    mean_s = np.mean(S)
    mean_v = np.mean(V)

    # Gabungkan fitur HSV dan GLCM
    features = [mean_s, mean_v]

    return np.array(features).reshape(1, -1)

# Parameter HSV untuk masking
h_range = (0, 179)
s_range = (25, 255)
v_range = (0, 255)

# Muat model SVM yang sudah disimpan
model = joblib.load('svm_model_2_fitur.joblib')

# Buka video
cap = cv2.VideoCapture('5cm_80RPM_biji campur.mp4')

# Ambil lebar dan tinggi dari video
frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

# Definisikan codec dan buat VideoWriter untuk menyimpan video dalam format MP4
fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # Codec yang digunakan untuk format .mp4
```

```
output_filename = 'output_klasifikasi_2_fitur.mp4'
fps = 20.0  # Frame per second dari video output
output_size = (frame_width, frame_height)  # Ukuran frame output, harus sama dengan video input

# Buat VideoWriter object
out = cv2.VideoWriter(output_filename, fourcc, fps, output_size)

while cap.isOpened():
    ret, frame = cap.read()
    if ret:
        # Proses cropping berdasarkan kontur yang terdeteksi dengan validasi ROI
        cropped_img, img_processed = detect_and_crop_contours(frame, h_range, s_range, v_range)
        # Ekstraksi fitur dari citra hasil cropping
        features = extract_hsv_glcm_features(cropped_img)
        # Cek apakah fitur ditemukan
        if features is not None:
            # Prediksi menggunakan model
            pred = model.predict(features)
            prob = model.predict_proba(features)
            # Map output prediksi ke label yang lebih deskriptif
            if pred[0] == 0:
                prediction_text = 'Biji Baik'
            elif pred[0] == 1:
                prediction_text = 'Biji Buruk'
            else:
                prediction_text = 'Tidak Diketahui'
            # Tampilkan prediksi pada frame
            prediction_prob = f"{np.max(prob) * 100:.2f}%"
            text = f'Prediksi: {prediction_text}, Probabilitas: {prediction_prob}'
            cv2.putText(img_processed, text, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        # Simpan frame yang sudah diproses ke file video
        out.write(img_processed)
        # Tampilkan frame di window
        cv2.imshow('Video', img_processed)

        # Keluar dengan menekan 'q'
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break
# Lepaskan capture, writer, dan tutup semua jendela
cap.release()
out.release()  # Jangan lupa untuk melepaskan VideoWriter setelah selesai
cv2.destroyAllWindows()
```