BIN : Bulletin Of Informatics
Volume 2, No. 1 Juli 2024
ISSN 3025-7417 (media online)
Hal 130-136

Deteksi Tingkat Kematangan Buah Tomat Menggunakan Pengolahan Citra Dengan OpenCV dan Python

Herlan Firdaus[1], Fikry Farizi[2], Roihan Aulia Syakur[3], Ariyanto Taufiq Ramadhan[4], Perani Rosyani[5]

Program Studi Teknik Informatika , Fakultas Ilmu Komputer
Universitas Pamulang, Jl. Raya Puspitek, Buaran, Kec. Pamulang, Kota Tangerang Selatan
Email: herlanfirdaus29@gmail.com[1], fahrizifikrie12@gmail.com [2], roihanaulia06@gmail.com [3], ariyantotauifqramadhan@gmail.com [3], dosen00837@unpam.ac.id [4]

Abstrak- Tomat adalah komoditas pertanian yang penting dan deteksi tingkat kematangannya berpengaruh besar pada kualitas dan nilai jual. Penelitian ini mengusulkan metode deteksi kematangan tomat menggunakan pengolahan citra digital dengan OpenCV dan Python. Sistem ini memanfaatkan konversi ruang warna RGB ke HSV, thresholding, dan operasi morfologi untuk menganalisis karakteristik visual tomat. Tahapannya mencakup input citra, pra-pemrosesan, konversi warna, segmentasi warna, thresholding, operasi morfologi, analisis piksel, klasifikasi kematangan, dan visualisasi hasil. Sistem ini mampu mengklasifikasikan tomat ke dalam kategori "Matang", "Setengah Matang", atau "Mentah" dengan akurasi tinggi. Penggunaan HSV efektif dalam segmentasi warna, dan operasi morfologi membantu mengurangi noise. Sistem ini berpotensi diimplementasikan dalam aplikasi sortasi otomatis dan quality control dalam industri pengolahan tomat.

Kata Kunci : Pengolahan citra digital,Deteksi kematangan tomat,OpenCV,Python, Segmentasi warna, Ruang warna HSV, Thresholding, Operasi morfologi, Analisis piksel, Klasifikasi kematangan.

Abstract-Tomatoes are an important agricultural commodity and detection of their ripeness level has a major influence on quality and selling value. This research proposes a method for detecting tomato ripeness using digital image processing with OpenCV and Python. This system utilizes RGB to HSV color space conversion, thresholding, and morphological operations to analyze the visual characteristics of tomatoes. The stages include image input, pre-processing, color conversion, color segmentation, thresholding, morphological operations, pixel analysis, maturity classification, and result visualization. The system is capable of classifying tomatoes into "Ripe", "Semi-Ripe" or "Unripe" categories with high accuracy. The use of HSV is effective in color segmentation, and morphological operations help reduce noise. This system has the potential to be implemented in automatic sorting and quality control applications in the tomato processing industry.

Keywords: Digital image processing, tomato ripeness detection, OpenCV, Python, color segmentation, HSV color space, thresholding, morphological operations, pixel analysis, ripeness classification.

1. PENDAHULUAN

Tomat merupakan salah satu komoditas pertanian yang penting dan banyak dikonsumsi di seluruh dunia. Tingkat kematangan tomat sangat mempengaruhi kualitas, rasa, dan nilai jualnya. Oleh karena itu, deteksi tingkat kematangan tomat yang akurat dan efisien sangat diperlukan dalam industri pertanian dan pengolahan makanan.

Pengolahan citra digital menawarkan solusi yang menjanjikan untuk mengatasi masalah ini. Dengan memanfaatkan teknologi computer vision, kita dapat mengembangkan sistem otomatis yang mampu mendeteksi tingkat kematangan tomat berdasarkan karakteristik visualnya. Pendekatan ini tidak hanya lebih cepat dan konsisten dibandingkan metode manual, tetapi juga dapat mengurangi kebutuhan tenaga kerja dan meningkatkan efisiensi dalam proses sortasi dan pengemasan.

Dalam penelitian ini, kami mengusulkan sebuah metode untuk mendeteksi tingkat kematangan buah tomat menggunakan pengolahan citra dengan memanfaatkan library OpenCV dan bahasa pemrograman Python. Metode ini menggabungkan beberapa teknik pengolahan citra, termasuk konversi ruang warna, thresholding, dan operasi morfologi, untuk menganalisis karakteristik visual tomat dan menentukan tingkat kematangannya.

2. METODOLOGI

BIN : Bulletin Of Informatics
Volume 2, No. 1 Juli 2024
ISSN 3025-7417 (media online)
Hal 130-136

Metodologi yang digunakan dalam penelitian ini terdiri dari beberapa tahap utama:

1. Input Citra:

Sistem menggunakan file gambar dari folder input serta mendukung format JPG, PNG, dan JPEG.

2. Pra-pemrosesan:

Citra diimpor menggunakan OpenCV kemudian dilakukan pengecekan ukuran gambar dan penskalaan jika diperlukan untuk memastikan ukuran minimum (300x200 piksel).

3. Konversi RGB ke HSV:

HSV lebih efektif untuk segmentasi warna karena memisahkan informasi warna (Hue) dari kecerahan (Value) dan saturasi.

4. Segmentasi Warna:

Didefinisikan beberapa range warna dalam ruang HSV untuk merepresentasikan berbagai tingkat kematangan tomat:

- Merah (2 range karena sifat siklik dari komponen Hue)
- Kuning-Oranye
- Hijau

5. Thresholding Gambar HSV:

Dilakukan thresholding untuk memisahkan area tomat berdasarkan warnanya, krusial untuk analisis kematangan.

6. Operasi Morfologi:

Mengurangi noise dan memperbaiki bentuk area yang terdeteksi, meningkatkan akurasi segmentasi.

7. Analisis Piksel:

- Dihitung jumlah piksel non-zero untuk masing-masing mask warna.
- Dihitung rasio piksel untuk setiap warna terhadap total piksel yang terdeteksi.

8. Klasifikasi Kematangan:

- Berdasarkan rasio piksel, tomat diklasifikasikan menjadi "Matang", "Setengah Matang", atau "Mentah".
- Jika tidak ada piksel yang terdeteksi, diklasifikasikan sebagai "Tidak terdeteksi".

9. Visualisasi Hasil:

- Hasil klasifikasi ditampilkan pada gambar asli menggunakan teks overlay.
- Digunakan background semi-transparan untuk meningkatkan keterbacaan teks.

3. IMPLEMENTASI DAN HASIL

3.1 Implementasi

Untuk implementasi kita dapat menggunakan bahasa pemrograman python dan pustaka OpenCV. Berikut kode yang dapat digunakan:

```
import cv2
import numpy as np
```

BIN : Bulletin Of Informatics
Volume 2, No. 1 Juli 2024
ISSN 3025-7417 (media online)
Hal 130-136

```
import os

def detect_tomato_ripeness(image_path):
    # Baca gambar
    image = cv2.imread(image_path)

    # Pastikan gambar memiliki ukuran minimum
    min_width, min_height = 300, 200
    h, w = image.shape[:2]
    if w < min_width or h < min_height:
        scale = max(min_width / w, min_height / h)
        image = cv2.resize(image, None, fx=scale, fy=scale, interpolation=cv2.INTER_CUBIC)

    # Konversi RGB ke HSV
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    # Definisikan range warna dalam HSV
    lower_red1 = np.array([0, 100, 100])
    upper_red1 = np.array([10, 255, 255])
    lower_red2 = np.array([160, 100, 100])
    upper_red2 = np.array([180, 255, 255])
    lower_yellow_orange = np.array([10, 100, 100])
    upper_yellow_orange = np.array([25, 255, 255])
    lower_green = np.array([35, 50, 50])
    upper_green = np.array([85, 255, 255])

    # Thresholding gambar HSV
    mask_red1 = cv2.inRange(hsv, lower_red1, upper_red1)
    mask_red2 = cv2.inRange(hsv, lower_red2, upper_red2)
    mask_red = cv2.bitwise_or(mask_red1, mask_red2)
    mask_yellow_orange = cv2.inRange(hsv, lower_yellow_orange, upper_yellow_orange)
    mask_green = cv2.inRange(hsv, lower_green, upper_green)

    # Morfologi untuk menghilangkan noise
    kernel = np.ones((5,5), np.uint8)
    masks = [mask_red, mask_yellow_orange, mask_green]
    for i in range(len(masks)):
```

BIN : Bulletin Of Informatics
Volume 2, No. 1 Juli 2024
ISSN 3025-7417 (media online)
Hal 130-136

```
        masks[i] = cv2.morphologyEx(masks[i], cv2.MORPH_OPEN, kernel)
        masks[i] = cv2.morphologyEx(masks[i], cv2.MORPH_CLOSE, kernel)

    # Hitung jumlah piksel untuk setiap warna
    red_pixels = cv2.countNonZero(masks[0])
    yellow_orange_pixels = cv2.countNonZero(masks[1])
    green_pixels = cv2.countNonZero(masks[2])

    # Tentukan tingkat kematangan
    total_pixels = red_pixels + yellow_orange_pixels + green_pixels
    if total_pixels > 0:
        red_ratio = red_pixels / total_pixels
        yellow_orange_ratio = yellow_orange_pixels / total_pixels
        green_ratio = green_pixels / total_pixels

        if red_ratio > 0.5:
            ripeness = "Matang"
        elif yellow_orange_ratio > 0.5 or (yellow_orange_ratio + red_ratio > 0.8 and
yellow_orange_ratio > red_ratio):
            ripeness = "Setengah Matang"
        else:
            ripeness = "Mentah"
    else:
        ripeness = "Tidak terdeteksi"

    # Tampilkan hasil
    text = f"Kematangan: {ripeness}"
    font = cv2.FONT_HERSHEY_SIMPLEX

    # Sesuaikan ukuran font berdasarkan ukuran gambar
    font_scale = min(w, h) / 500  # Sesuaikan nilai pembagi untuk mengubah ukuran relatif font
    font_thickness = max(1, int(min(w, h) / 300))  # Sesuaikan nilai pembagi untuk mengubah
ketebalan font

    text_size = cv2.getTextSize(text, font, font_scale, font_thickness)[0]

    # Posisi teks yang dinamis
```

BIN : Bulletin Of Informatics
Volume 2, No. 1 Juli 2024
ISSN 3025-7417 (media online)
Hal 130-136

```
    text_x = 10
    text_y = text_size[1] + 10

    # Gambar background semi-transparan
    overlay = image.copy()
    cv2.rectangle(overlay, (text_x - 5, text_y - text_size[1] - 5),
                  (text_x + text_size[0] + 5, text_y + 5), (0, 0, 0), -1)
    cv2.addWeighted(overlay, 0.6, image, 0.4, 0, image)

    # Gambar teks utama
    cv2.putText(image, text, (text_x, text_y), font, font_scale, (255, 255, 255), font_thickness)

    return image, ripeness

def process_images_in_folder(input_folder, output_folder):
    # Buat folder output jika belum ada
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    # Dapatkan semua file gambar dalam folder input
    image_files = [f for f in os.listdir(input_folder) if f.lower().endswith(('.png', '.jpg', '.jpeg'))]

    results = {}

    for image_file in image_files:
        input_path = os.path.join(input_folder, image_file)
        processed_image, ripeness = detect_tomato_ripeness(input_path)

        # Simpan hasil
        results[image_file] = ripeness

        # Simpan gambar yang telah diproses di folder output
        output_path = os.path.join(output_folder, f"processed_{image_file}")
        cv2.imwrite(output_path, processed_image)

        print(f"Processed {image_file}: {ripeness}")
```

BIN : Bulletin Of Informatics
Volume 2, No. 1 Juli 2024
ISSN 3025-7417 (media online)
Hal 130-136

```
    return results


# Gunakan fungsi
input_folder = "input"
output_folder = "output"
results = process_images_in_folder(input_folder, output_folder)


# Tampilkan ringkasan hasil
print("\nRingkasan Hasil:")
for image, ripeness in results.items():
    print(f"{image}: {ripeness}")
```

3.2 Hasil

1. Sistem berhasil mengklasifikasikan tomat ke dalam tiga kategori kematangan: Matang, Setengah Matang, dan Mentah.

2. Penggunaan ruang warna HSV efektif dalam membedakan berbagai tingkat kematangan tomat berdasarkan warnanya.

3. Operasi morfologi berhasil mengurangi noise dan memperbaiki area deteksi.

4. Sistem dapat menangani variasi ukuran gambar input dengan melakukan penskalaan otomatis jika diperlukan.

5. Visualisasi hasil dengan teks overlay dan background semi-transparan memberikan feedback yang jelas dan mudah dibaca.

Gambar 1. Hasil Deteksi Tingkat Kematangan Buah Tomat

4. KESIMPULAN

Penelitian ini mendemonstrasikan efektivitas penggunaan teknik pengolahan citra digital dengan OpenCV dan Python dalam mendeteksi tingkat kematangan buah tomat dari file gambar. Metode yang diusulkan, yang menggabungkan konversi ruang warna, thresholding berbasis HSV,

BIN : Bulletin Of Informatics
Volume 2, No. 1 Juli 2024
ISSN 3025-7417 (media online)
Hal 130-136

operasi morfologi, dan analisis rasio piksel, terbukti mampu mengklasifikasikan tomat ke dalam berbagai tingkat kematangan dengan cara yang efisien.

Sistem ini memiliki potensi untuk diimplementasikan dalam aplikasi sortasi otomatis atau quality control dalam industri pengolahan tomat. Kelebihannya termasuk kemampuan untuk bekerja dengan file gambar yang sudah ada, fleksibilitas dalam menangani berbagai ukuran gambar, dan visualisasi hasil yang informatif.

REFERENSI

S. Arivazhagan, S. S. Shebiah, R.N., Nidhyanandhan, and L. Ganesan, “Fruit recognition using color and texture features,” J. Emerg. Trends Comput. Inform. Sci, vol. 1, pp. 90–94, 2010.

M. R. Satpute and S. M. Jagdale, “Color, size, volume, shape and texture feature extraction techniques for fruits: a review,” Int. Res. J. Eng. Technol., vol. 3, pp. 703–708, 2016.

Tarigan, Nirma Yopita Sari, Utama, I Made Supartha, and Kencana, Pande K Diah, 'Memperthankan Mutu Buah Tomat Segar Dengan Pelapisan Minyak Nabati'. (2015).

Zulkhaidi, T. C. A. S., Maria, E., & Yulianto, Y. (2020). Pengenalan Pola Bentuk Wajah dengan OpenCV. Jurnal Rekayasa Teknologi Informasi (JURTI), 3(2), 181.

Rendy Pratama, Achmad Fuad Assagaf, Firman Tempola, "DETEKSI KEMATANGAN BUAH TOMAT BERDASARKAN FITUR WARNA MENGGUNAKAN METODE TRANSFORMASI RUANG WARNA HIS," Jurnal Informatika dan Ilmu Komputer (JIKO), vol. 2, no. 2, pp. 81-86, Oktober 2019.

Napitu, S., Paramita Panjaitan, R., Nulhakim, P.A., & Khalik Lubis, M. (2023). Klasifikasi Buah Jeruk Segar dan Busuk Berdasarkan RGB dan HSV Menggunakan Metode KNN. Jurnal SAINTEKOM.

Rosyani, P., & Priambodo, J. (2019). Penilaian Kinerja Karyawan Berprestasi Dengan Metode Simple Additive Weighting. *International Journal of Artificial Intelligence*, *6*(1), 82-111.

Ardiansyah, M. F., & Rosyani, P. (2023). Perancangan UI/UX Aplikasi Pengolahan Limbah Anorganik Menggunakan Metode Design Thinking. *LOGIC: Jurnal Ilmu Komputer dan Pendidikan*, *1*(4), 839-853.

Supriyadi, D., Fajar, F. R., Utami, M., Nurjanah, S., Restiani, A., Sari, Y. P., & Rosyani, P. (2022). Analisis Sistem Pakar Diagnosa Penyakit THT Menggunakan Metode Certainty Factor. *OKTAL: Jurnal Ilmu Komputer dan Sains*, *1*(06), 652-657.