Laporan Praktikum Kontrol Cerdas

Minggu ke-2

Nama : Amalia Dwi Nurahma

NIM : 224308075

Kelas : TKA 7D

Akun Github (Tautan) : https://github.com/amaliadwinurahma

Student Lab Assistant :

1. Judul Percobaan

Machine Learning for Control Systems

2. Tujuan Percobaan

Tujuan dari praktikum minggu kedua ini adalah:

a. Memahami penerapan algoritma Machine Learning (K-Nearest Neighbor) untuk klasifikasi warna dasar.
b. Menggunakan metode deteksi warna berbasis ruang warna HSV dengan bantuan OpenCV.
c. Membandingkan pendekatan berbasis klasifikasi (KNN) dengan threshold HSV dalam mengenali warna.
d. Melatih kemampuan membuat bounding box dan menampilkan label warna secara real-time dari kamera.

3. Landasan Teori

A. K-Nearest Neighbor (KNN)

KNN merupakan algoritma supervised learning yang digunakan untuk klasifikasi maupun regresi. Konsepnya adalah menentukan kelas sebuah data uji berdasarkan mayoritas kelas dari tetangga terdekatnya dalam ruang fitur. Pada percobaan ini, KNN digunakan untuk mengklasifikasikan warna berdasarkan nilai RGB yang sudah dinormalisasi (Han et al., 2012).

B. Ruang Warna HSV

HSV (Hue, Saturation, Value) lebih sesuai digunakan untuk deteksi warna dibanding RGB karena lebih menyerupai cara manusia mengenali warna. Hue menunjukkan jenis warna, Saturation menyatakan tingkat kejenuhan, sedangkan Value mewakili kecerahan. OpenCV menyediakan fungsi cv2.inRange() untuk memisahkan objek dengan rentang HSV tertentu (Szeliski, 2010).

C. Computer Vision dengan OpenCV

OpenCV adalah pustaka open-source yang banyak digunakan dalam pengolahan citra digital. Dengan OpenCV, proses konversi ruang warna, pembuatan masking, deteksi kontur, hingga visualisasi bounding box dapat dilakukan dengan mudah dan efisien (Putra, 2020).

D. Integrasi KNN dan HSV untuk Deteksi Warna

Pendekatan gabungan ini digunakan untuk meningkatkan akurasi sistem. Deteksi HSV dapat langsung mengenali warna dengan threshold, sedangkan KNN membantu mengklasifikasikan warna ketika kondisi cahaya atau nilai HSV ambigu.

4. Analisis dan Diskusi

A. Analisis

Program pada minggu ke-2 dirancang untuk mendeteksi warna menggunakan dua pendekatan:

1) **Metode HSV** → mendeteksi warna dengan thresholding berdasarkan nilai Hue. Cocok untuk warna-warna dasar (merah, hijau, biru, kuning, cyan, magenta, hitam, putih).
2) **Metode KNN** → melakukan klasifikasi berdasarkan dataset RGB sederhana yang terdiri dari delapan warna. Data dilatih menggunakan StandardScaler agar fitur lebih seimbang.

Hasil prediksi dari HSV maupun KNN kemudian divisualisasikan dalam bentuk bounding box dan label teks pada objek di tengah frame kamera. Selain itu, ditampilkan juga tingkat akurasi model KNN.

B. Diskusi

1) Metode HSV lebih cepat, tetapi rentan terhadap perubahan pencahayaan.
2) Metode KNN memberikan fleksibilitas dan dapat dikembangkan untuk dataset warna lebih banyak.
3) Perpaduan keduanya menjadikan sistem lebih adaptif: HSV untuk deteksi langsung, KNN untuk fallback saat warna sulit dikenali.

5. Assignment

Assignment pada praktikum minggu kedua adalah mengembangkan program supaya :

a. Dapat mengenali lebih banyak warna (8 warna: merah, hijau, biru, kuning, cyan, magenta, hitam, putih).
b. Menambahkan klasifikasi warna berbasis Machine Learning (KNN).
c. Membuat fungsi tambahan untuk menggambar bounding box dinamis yang berisi label warna serta tingkat kepercayaan (confidence score).
d. Menampilkan akurasi model KNN di layar secara real-time.

6. Data dan Output Hasil Pengamatan

A. Dataset Warna (untuk KNN)

Warna	R	G	B
Red	255	0	0
Green	0	255	0
Blue	0	0	255
Yellow	255	255	0
Cyan	0	255	255
Magenta	255	0	255
Black	0	0	0
White	255	255	255

B. Coding

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import cv2

# Dataset contoh untuk training KNN
data = {
    'ColorName': ['Red', 'Green', 'Blue', 'Yellow', 'Cyan', 'Magenta', 'Black', 'White'],
    'R': [255, 0, 0, 255, 0, 255, 0, 255],
    'G': [0, 255, 0, 255, 255, 0, 0, 255],
    'B': [0, 0, 255, 0, 255, 255, 0, 255]
}
color_data = pd.DataFrame(data)

X = color_data[['R', 'G', 'B']].values
y = color_data['ColorName'].values

# Normalisasi fitur RGB
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)

knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred) * 100
print(f'Akurasi model KNN: {accuracy:.2f}%')

# Warna bounding box sesuai warna deteksi (BGR)
color_bgr_map = {
    "Red": (0, 0, 255),
    "Yellow": (0, 255, 255),
    "Green": (0, 255, 0),
    "Blue": (255, 0, 0),
    "Cyan": (255, 255, 0),
    "Magenta": (255, 0, 255),
    "Black": (0, 0, 0),
    "White": (255, 255, 255)
}

def detect_color_hsv(hsv_pixel):
    # Red
    lower_red1 = np.array([0, 120, 70])
    upper_red1 = np.array([10, 255, 255])
    lower_red2 = np.array([160, 120, 70])
    upper_red2 = np.array([179, 255, 255])
    # Yellow
    lower_yellow = np.array([20, 100, 100])
    upper_yellow = np.array([30, 255, 255])
    # Green
    lower_green = np.array([40, 50, 50])
    upper_green = np.array([80, 255, 255])
    # Blue
    lower_blue = np.array([90, 50, 50])
    upper_blue = np.array([130, 255, 255])
```

```
    # Cyan
    lower_cyan = np.array([81, 50, 50])
    upper_cyan = np.array([100, 255, 255])
    # Magenta
    lower_magenta = np.array([140, 50, 50])
    upper_magenta = np.array([160, 255, 255])
    # White (rendah saturasi, tinggi value)
    lower_white = np.array([0, 0, 200])
    upper_white = np.array([179, 40, 255])
    # Black (rendah value)
    lower_black = np.array([0, 0, 0])
    upper_black = np.array([179, 255, 50])

    mask_red = cv2.inRange(hsv_pixel, lower_red1, upper_red1) +
cv2.inRange(hsv_pixel, lower_red2, upper_red2)
    mask_yellow = cv2.inRange(hsv_pixel, lower_yellow, upper_yellow)
    mask_green = cv2.inRange(hsv_pixel, lower_green, upper_green)
    mask_blue = cv2.inRange(hsv_pixel, lower_blue, upper_blue)
    mask_cyan = cv2.inRange(hsv_pixel, lower_cyan, upper_cyan)
    mask_magenta = cv2.inRange(hsv_pixel, lower_magenta, upper_magenta)
    mask_white = cv2.inRange(hsv_pixel, lower_white, upper_white)
    mask_black = cv2.inRange(hsv_pixel, lower_black, upper_black)

    if mask_red[0] > 0:
        return "Red"
    elif mask_yellow[0] > 0:
        return "Yellow"
    elif mask_green[0] > 0:
        return "Green"
    elif mask_blue[0] > 0:
```

```
        return "Blue"
    elif mask_cyan[0] > 0:
        return "Cyan"
    elif mask_magenta[0] > 0:
        return "Magenta"
    elif mask_white[0] > 0:
        return "White"
    elif mask_black[0] > 0:
        return "Black"
    else:
        return None

def draw_label_and_box(image, text, pos, box_size, color, accuracy_score=None):
    font = cv2.FONT_HERSHEY_SIMPLEX
    font_scale = 1.0
    thickness = 2
    text_size, _ = cv2.getTextSize(text, font, font_scale, thickness)
    while (text_size[0] > box_size or text_size[1] > box_size//2) and font_scale >
0.1:
        font_scale -= 0.1
        text_size, _ = cv2.getTextSize(text, font, font_scale, thickness)
    x, y = pos
    top_left = (x - box_size // 2, y - box_size // 2)
    bottom_right = (x + box_size // 2, y + box_size // 2)
    cv2.rectangle(image, top_left, bottom_right, color, 2)
    text_x = x - text_size[0] // 2
    text_y = y - box_size // 2 - 10
    if text_y < 0:
        text_y = y + box_size // 2 + text_size[1] + 10
    overlay = image.copy()
```

```
    cv2.rectangle(overlay, (text_x - 5, text_y - text_size[1] - 5),
                  (text_x + text_size[0] + 5, text_y + 5), color, -1)
    alpha = 0.5
    cv2.addWeighted(overlay, alpha, image, 1 - alpha, 0, image)
    brightness = (color[0]*0.299 + color[1]*0.587 + color[2]*0.114)
    text_color = (255, 255, 255) if brightness < 128 else (0, 0, 0)
    cv2.putText(image, text, (text_x, text_y), font, font_scale, text_color, thickness)
    if accuracy_score is not None:
        accuracy_text = f"Accuracy: {accuracy_score:.1f}%"
        acc_text_size, _ = cv2.getTextSize(accuracy_text, font, font_scale * 0.7,
thickness - 1)
        acc_text_x = x - acc_text_size[0] // 2
        acc_text_y = text_y + text_size[1] + 20
        cv2.rectangle(overlay, (acc_text_x - 5, acc_text_y - acc_text_size[1] - 5),
                      (acc_text_x + acc_text_size[0] + 5, acc_text_y + 5), (50, 50, 50), -1)
        cv2.addWeighted(overlay, alpha, image, 1 - alpha, 0, image)
        cv2.putText(image, accuracy_text, (acc_text_x, acc_text_y), font,
                    font_scale * 0.7, (255, 255, 255), thickness - 1)

def get_prediction_confidence(pixel_rgb_scaled):
    probabilities = knn.predict_proba(pixel_rgb_scaled)
    max_prob = np.max(probabilities) * 100
    return max_prob

cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Error membuka kamera")
    exit()

while True:
```

```
    ret, frame = cap.read()
    if not ret:
        break

    height, width, _ = frame.shape
    center_x, center_y = width // 2, height // 2
    pixel_bgr = frame[center_y, center_x].reshape(1, 1, 3)
    pixel_hsv = cv2.cvtColor(pixel_bgr, cv2.COLOR_BGR2HSV)

    color_special = detect_color_hsv(pixel_hsv)
    confidence = accuracy

    if color_special:
        color_pred = color_special
        confidence = 90.0
    else:
        pixel_rgb = pixel_bgr[0][0][::-1].reshape(1, -1)
        pixel_rgb_scaled = scaler.transform(pixel_rgb)
        color_pred = knn.predict(pixel_rgb_scaled)[0]
        confidence = get_prediction_confidence(pixel_rgb_scaled)
    box_color = color_bgr_map.get(color_pred, (0, 0, 0))
    draw_label_and_box(frame, color_pred, (center_x, center_y), 60, box_color,
confidence)
    cv2.putText(frame, f"Model Accuracy: {accuracy:.1f}%", (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
    cv2.imshow('Deteksi Warna dengan Bounding Box Dinamis', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
```

```
cv2.destroyAllWindows()
```

C. Analisis Coding

1) Pembuatan Dataset Warna

```
data = {
   'ColorName': ['Red', 'Green', 'Blue', 'Yellow', 'Cyan', 'Magenta', 'Black', 'White'],
   'R': [255, 0, 0, 255, 0, 255, 0, 255],
   'G': [0, 255, 0, 255, 255, 0, 0, 255],
   'B': [0, 0, 255, 0, 255, 255, 0, 255]
}
```

a. Dataset terdiri dari **8 warna dasar** dengan representasi nilai **RGB**.
b. Data ini akan digunakan untuk melatih model KNN agar bisa mengenali warna berdasarkan komposisi RGB.

2) Normalisasi dan Training Model KNN

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)


knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
```

a. **StandardScaler** dipakai untuk menormalkan nilai RGB agar distribusinya lebih seimbang.
b. Dataset dibagi menjadi **data latih** dan **data uji** dengan perbandingan 80:20.
c. Model **KNN dengan k=3** dilatih untuk mengklasifikasikan warna.

3) Evaluasi Model

```
y_pred = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred) * 100
print(f'Akurasi model KNN: {accuracy:.2f}%')
```

a. Model dievaluasi menggunakan **akurasi** terhadap data uji.

b. Nilai akurasi ini akan ditampilkan di layar saat program dijalankan.

4) Deteksi Warna dengan HSV

def detect_color_hsv(hsv_pixel): definisi rentang HSV untuk warna merah, hijau, biru, kuning, cyan, magenta, putih, hitam

a. Fungsi ini menggunakan cv2.inRange() untuk memeriksa apakah sebuah piksel masuk ke rentang HSV tertentu.
b. Jika masuk, fungsi mengembalikan nama warna (contoh: "Red", "Green").
c. HSV dipakai karena lebih stabil terhadap perubahan cahaya dibanding RGB.

5) Visualisasi Bounding Box

```
def draw_label_and_box(image, text, pos, box_size, color, accuracy_score=None):
    # menggambar kotak (bounding box)
    # menambahkan label nama warna
    # menampilkan confidence/akurasi
```

a. Fungsi ini menggambar kotak di sekitar titik pusat frame kamera.
b. Warna kotak sesuai warna yang terdeteksi (color_bgr_map).
c. Label teks ditambahkan di atas kotak dengan background semi-transparan.
d. Jika ada nilai akurasi, teks *accuracy* juga ditampilkan.

6) Looping Kamera Real-Time

```
cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    cv2.imshow('Deteksi Warna dengan Bounding Box Dinamis', frame)
```

a. Program membuka kamera laptop/PC.
b. Pada setiap frame:
    1) Ambil **piksel tengah frame** sebagai sampel warna.
    2) Coba deteksi dengan metode **HSV** terlebih dahulu.
    3) Jika gagal, gunakan prediksi **KNN**.
    4) Tampilkan hasil deteksi dengan bounding box + label.

7) Integrasi HSV & KNN

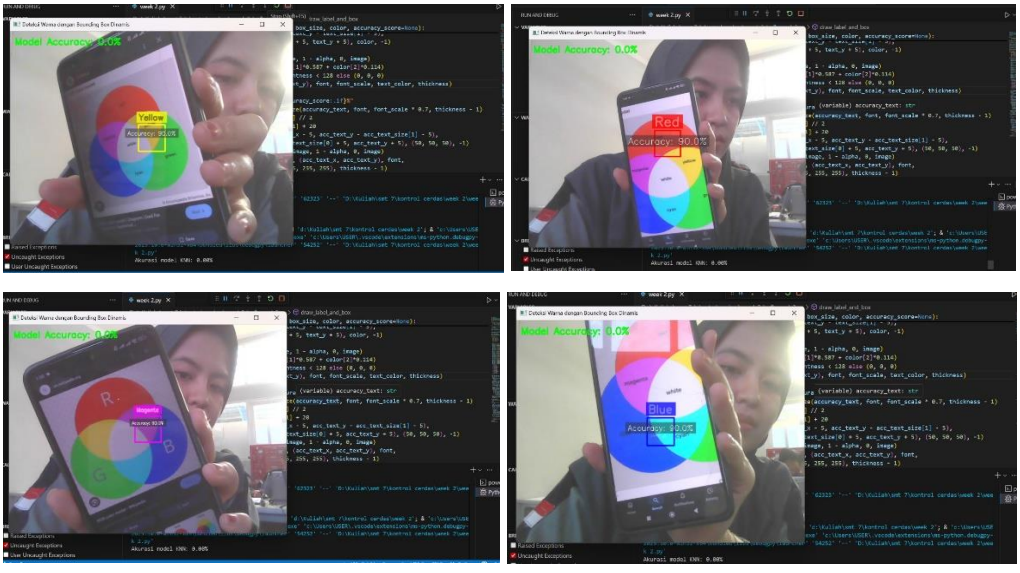
```
if color_special:
    color_pred = color_special
    confidence = 90.0
else:
    pixel_rgb = pixel_bgr[0][0][::-1].reshape(1, -1)
    pixel_rgb_scaled = scaler.transform(pixel_rgb)
    color_pred = knn.predict(pixel_rgb_scaled)[0]
    confidence = get_prediction_confidence(pixel_rgb_scaled)
```

a. Prioritas utama deteksi warna dengan HSV karena lebih cepat.
b. Jika HSV gagal, sistem menggunakan prediksi dari KNN.
c. Confidence KNN diambil dari probabilitas prediksi (predict_proba).

D. Output

1) Program berhasil menampilkan bounding box dengan label warna.
2) Tingkat akurasi KNN ditampilkan di layar.
3) Warna terdeteksi dengan baik pada kondisi cahaya stabil.

Output program ini terlihat seperti pada gambar di bawah ini :



7. Kesimpulan

Dari praktikum ini dapat disimpulkan bahwa:

a. KNN dapat digunakan untuk klasifikasi warna dengan dataset sederhana, dan hasilnya cukup akurat.
b. Deteksi berbasis HSV lebih sederhana namun kurang adaptif terhadap perubahan pencahayaan.
c. Integrasi KNN dan HSV menghasilkan sistem deteksi warna real-time yang lebih handal.
d. Bounding box dinamis dan label teks mempermudah visualisasi hasil deteksi.

8. Saran

a. Melakukan kalibrasi nilai HSV secara otomatis untuk mengurangi pengaruh pencahayaan.
b. Menambah jumlah data latih agar model KNN lebih robust.
c. Menggunakan kamera dengan resolusi lebih baik untuk meningkatkan kualitas citra.
d. Mengembangkan program agar dapat mendeteksi bentuk objek sekaligus selain warna.

9. Daftar Pustaka

Han, J., Kamber, M. & Pei, J. (2012). *Data Mining: Concepts and Techniques*. 3rd edn. Morgan Kaufmann.

Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer.

Putra, R. (2020). *Pengolahan Citra Digital dengan Python dan OpenCV*. Yogyakarta: Andi Offset.