DOKUMENTASI PROYEK MINI: PERMAINAN MENCARI WARNA DENGAN MENGGUNAKAN OPENCV PYTHON

1. Pendahuluan

1.1 Tujuan dan Gambaran Permainan

Tujuan proyek ini dibuat adalah sebagai pendalaman dan pembiasaan untuk menggunakan salah satu *library computer vision* yaitu, OpenCV Python. Dari proyek sederhana ini *output* utamanya ialah pemahaman akan berbagai fitur dasar OpenCV dan implementasinya dalam sebuah kode. Serta salah satu syarat untuk tes lanjutan ROBOTIIK FILKOM UB.

Gambaran dari permainan mencari warna ini adalah pemain ditugaskan untuk mencari dan menggunkaan benda yang ada disektiar mereka atau bahkan tubuh mereka seperti, botol minum, rambut, dan lain-lain, yang sesuai dengan warna yang telah ditentukan oleh program.

1.2 OpenCV Python

OpenCV (Open Source Computer Vision) adalah *library* open-source untuk Python yang dirancang khusus untuk kebutuhan *computer vision* dan *machine learning*. Dengan lebih dari 2.500 algoritma yang telah dioptimalkan, OpenCV memungkinkan pengolahan citra dan video secara efisien, sehingga mempercepat pengembangan berbagai produk komersial.

OpenCV merupakan proyek yang diinisiasi oleh Intel pada tahun 1999 oleh Gary Bradsky dan dirilis secara publik pada tahun 2000. Salah satu pencapaiannya yang signifikan adalah penggunaannya dalam *Stanley*, kendaraan otonom yang berhasil memenangkan *2005 DARPA Grand Challenge*. Hingga saat ini, OpenCV terus berkembang dan diperbarui, menjadikannya salah satu pustaka utama dalam bidang *computer vision* dan kecerdasan buatan.

1.3 Library yang Digunakan

Untuk menunjang proyek kecil ini, perlu beberapa *library* yang akan membantu proyek ini, diantaranya :

1.3.1 Numpy

Numpy adalah sebuah *library open source* python yang memiliki fungsi utama untuk komputasi ilmiah. *Library* ini dikenal dengan kecepatan dan efisiensinya dalam menghitung suatu *array*. Dalam beberapa fungsi OpenCV diperlukan nilai *array* dalam format Numpy. Itulah mengapa dalam proyek ini, Numpy merupakan salah satu *library* yang perlu di-*import*.

Instalasi Numpy :

```
pip install numpy
```

```
*atau dapat merujuk pada : https://numpy.org/install/
```

1.3.2 OpenCV

Library utama dalam proyek ini. Berguna dalam pembacaan webcam pada perangkat dan mendeteksi kontur terutama pada benda yang terdeteksi memiliki warna yang dicari.

Instalasi OpenCV :

```
pip intall opencv-python
```

1.3.3 Random

Library bawaan dari python, berfungsi untuk mengacak warna yang harus dicari oleh pemain sehingga tidak dalam urutan yang sama.

1.3.4 Time

Library bawaan dari python, berfungsi untuk mencatat waktu saat benda berhasil terdeteksi sehingga dapat melakukan perhitungan semacam *timer* untuk sela pergantian waktu nantinya, sehingga pemain dapat bersiap-siap terlebih dahulu.

2. Penjelasan Kode

Bagian ini menjelaskan struktur kode yang digunakan dalam proyek ini. Kode terdiri dari *import library* seusai yang ada di pendahuluan, pembacaan webcam pemain, inisialisasi variabel utama, dan logika utama dalam permainan.

2.1 Import Library yang Digunakan

```
import cv2 as cv
import numpy as np
import random
import time
```

2.2 Membaca Webcam

```
cap = cv.VideoCapture(0)
```

Untuk membaca kamera pengguna, diperlukan fungsi **`cv.VideoCapture()`** sebagai awal untuk pendeteksian yang disimpan dalam variabel beranama **`cap`** . Terlihat bahwa nilai yang akan dibaca oleh OpenCV bernilai “0” atau nol yang artinaya akan membaca kamera bawaan dari laptop pemain. Jika ingin menggunkaan kamera eksternal (pastikan sudah terdeteksi oleh laptop anda), dapat memasukan nilai “1”, “2”, “3”, dan

seterusnya sampai nilai itu tepat membaca webcam anda. Dapat juga membuat kode pengecek apakah kamera sudah terdeteksi atau belum seperti ini.

```
if not cap.isOpened():
   print("Tidak ada kamera yang terdeteksi!")
   exit(0)
```

Kode di atas akan memeriksa dengan cara menggunakan fungsi **`isOpened()`** yang akan mengembalikan nilai boolean, jika bernilai *False*, maka tidak ada kamera yang terbaca oleh OpenCV sehingga dapat langusng keluar dari program.

2.3 Insialisasi Variabel Utama

```
colors_ranges = {
    "merah": ([0, 100, 100], [10, 255, 255]),
    "oranye": ([11, 100, 100], [25, 255, 255]),
    "kuning": ([26, 100, 100], [35, 255, 255]),
    "hijau": ([36, 100, 100], [85, 255, 255]),
    "biru": ([96, 100, 100], [130, 255, 255]),
    "putih": ([0, 0, 200], [180, 30, 255]),
    "hitam": ([0, 0, 0], [180, 255, 50]),
}

colors_list = [i for i in colors_ranges.keys()]
target_color = colors_list[random.randint(0, 100) % 7]
already_detected_color = []
tidak_memiliki_warna = []
found = False
start_time = None
next_color = None
```

Terdapat 5 variabel utama yang akan digunakan, mengapa tidak di dalam sebuah loop? Karena variabel ini bersifat statis atau tidak perlu dirubah di setiap iterasinya jika tidak diinginkan. Terutama pada variabel **`target_color`** yang memanfaatkan **`random`** , jika terdapat dalam loop, untuk setiap iterasi (dalam konteks ini setiap frame dari kamera) maka nilainya akan berubah. Sehingga, tidak memberikan pemain kesempatan untuk mencari barang.

Berikut penjelasan masing-masing variabel:

- **colors_ranges**
  Python *Dictionary* yang bertugas untuk menyimpan nama warna dan juga nilai warna dalam format HSV (*Hue Saturation Value*). Mengapa dalam HSV? Karena jika video warna disimpan dalam format RGB atau BGR, maka warna dapat berubah drastic seusai dengan kondisi cahaya, maka untuk memastikan nilainya konstan, OpenCV merekomendasikan untuk mengubah format warnanya ke HSV. Terdapat *tuples* yang berisikan 2 *list* yaitu, batas bawah warna dan batas atas warna yang nantinya dapat dijadikan sebagai *range* warna.
- **colors_list**

Python *list* yang menyimpan nama-nama dari warna.

- **target_color**
  Bertugas menyimpan nama dari warna yang ingin dideteksi dari benda yang ada.
- **already_detected_color**
  List yang bertugas untuk menyimpan warna yang sudah pernah dideteksi, sehingga permainan tidak akan berulang dalam loop yang warnanya tersisa warna yang sama.
- **tidak_memiliki_warna**
  List yang berguna untuk menyimpan warna yang sekiranya pemain tidak dapat jangkau.
- **found**
  Sebuah inisialisasi, secara *default* akan bernilai *False* karena belum ada benda dengan warna yang sesuai yang terdeteksi, yang nantinya akan bernilai *True* saat benda dengan warna yang sesuai terdeteksi.
- **start_time**
  Inisialisasi variabel pencatat waktu saat bend itu ditemukan, dalam logika utama nantinya akan digunakan untuk menghitung waktu sejak bend itu ditemukan sehingga dapat menerapkan sebuah *timer* jeda waktu sebelum warna selanjutnya.
- **next_color**
  Menyimpan warna selanjutnya yang perlu dicari. Bernilai awal *None* yang nantinya akan diisi. Berbeda dengan `target_color`, variabel ini akan menggantikan posisi `target_color` dalam loop, sehingga variabel `target_color` hanya bertugas untuk menyimpan saja.

2.4 Logika Utama

```
while cap.isOpened():
    # Membaca webcam
    success, frame = cap.read()

    # Mengatur ulang ukuran webcam yang akan ditampilkan
    resize = cv.resize(frame, (1280, 720))

    if not success:
        print("Tidak dapat merender frame!")
        break

    # Merubah format warna dari BGR ke HSV
    hsv_frame = cv.cvtColor(resize, cv.COLOR_BGR2HSV)

    # Menyimpan warna yang tersisa dan selalu update warna yang tersisa
    colors_remaining = [color for color in colors_list if color not in already_detected_color and color
not in tidak_memiliki_warna]

    # Menyimpan nilai array pertama dan kedua dari dictionary
    lower_value, upper_value = np.array(colors_ranges[target_color][0]),
np.array(colors_ranges[target_color][1])

    # Masking
    mask = cv.inRange(hsv_frame, lower_value, upper_value)
```

```
    masking_result = cv.bitwise_and(resize, resize, mask=mask)

    if target_color:
        cv.putText(resize, f"Find = {target_color}", (50, 50), cv.FONT_HERSHEY_COMPLEX, 1.1, (255, 0, 0),
2, cv.LINE_AA)
    else:
        cv.putText(resize, "Semua warna sudah pernah terdeteksi", (50, 50),
cv.FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 0, 255), 2, cv.LINE_AA)
        break

    # Mendeteksi batas-batas (kontur) dari objek yang ada dalam gambar biner (mask).
    contours, hierarchy = cv.findContours(mask, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)

    # Jika terdapat kontur
    if contours:
        for contour in contours:
            area = cv.contourArea(contour) # Hitung luas kontur
            if area > 1000: #Jika luas benda lebih dari 1000px

                # Menggambar semua kontur yang ada di dalam frame
                cv.drawContours(resize, contours, -1, (0, 255, 0), 3)
                cv.putText(resize, "Warna Ditemukan!", (1280 - 250, 50), cv.FONT_HERSHEY_COMPLEX, 1, (0,
255, 0), 2, cv.LINE_AA)

                # Warna sesuai (sudah ditemukan)
                found = True
                already_detected_color.append(target_color)

                # Catat waktu awal saat berhasil ditemukan
                start_time = time.time()

    if found and start_time is not None:
        # Hitung waktu yang sudah jalan sejak waktu awal dicatat
        elapsed_time = time.time() - start_time

        # Waktu tersisa sebelum berganti secara otomatis ke warna yang lain
        remaining_time = max(10 - int(elapsed_time), 0)
        cv.putText(resize, f"Next color in: {remaining_time}s", (50, 100), cv.FONT_HERSHEY_COMPLEX, 1,
(255, 255, 255), 2, cv.LINE_AA)

        # Jika waktu yang dijalani lebih dari 10 detik, maka akan mengubah warna yang harus dicari dan
mereset beberapa kondisi
        if elapsed_time >= 10:
            # Akan berubah warna hanya jika masih ada warna yang tersisa
            if len(colors_remaining) > 0:
                next_color = random.choice(colors_remaining)
                target_color = next_color
            else:
                print("tidak ada warna tersisa")
                break

            # Reset kondisi ditemukannya warna
            found = False

            # Reset waktu ditemukannya warna
            start_time = None

    # Menampilkan frame webcam
    cv.imshow("Cari Warna", resize)

    # Menampilkan frame hasil masking
    cv.imshow("Hasil Masking", masking_result)

    # Jika pemain stuck dan tidak dapat menemukan benda yang sesuai warnanya, maka dapat menekan tombol n
untuk pilihan warna yang lain
    if cv.waitKey(1) & 0xFF == ord("n"):
        tidak_memiliki_warna.append(target_color)
        # Hanya akan berubah warna hanya jika masih ada warna tersisa
```

```
        if len(colors_remaining) > 0:
            next_color = random.choice(colors_remaining)
            target_color = next_color
        else:
            print("Tidak ada warna tersisa")
            break

    # Keluar dari program
    elif cv.waitKey(1) & 0xFF == ord("q"):
        break

cap.release()
cv.destroyAllWindows()
```

Kode di atas merupakan bagian yang bertanggung jawab dalam pendeteksian warna pada objek.

Dimulai dari **`while cap.isOpened()`** atau selama kamera terbuka, bagian awal yang akan program ini lakukan adalah membaca setiap frame dalam kamera dan merubah ukuran *windows* yang nantinya akan tampil. Terdapat juga baris kode pengecekan apakah kamera dapat terender dengan baik atau tidak, jika tidak maka akan langsung keluar dari *loop* dan program akan berakhir.

Baris kode berikutnya memiliki kontribusi terbesar dalam logika utama permainan ini. Dimulai dari merubah *colorspace* atau ruang warna dari kamera yang awalnya BGR menjadi HSV. Yang kemudian dilanjutkan dengan variabel **`colors_remainig`** yang berupa *list* untuk menyimpan warna apa saja yang tersisa untuk dideteksi, jika warna tersebut belum pernah dideteksi dan tidak dalam list **`tidak_memiliki_warna`**. Variabel ini akan terus berubah karena di dalam sebuah loop atau mudahnya akan selalu *update* nilai dari terbaru. Selanjutnya, terdapat dua variabel **`lower_value`** dan **`upper_value`**, yang bertugas untuk menyimpan nilai HSV masing-masing warna dalam format array Numpy. Setelahnya, ada proses *masking* agar kamera hanya mendeteksi warna sesuai dengan nilai jangkauan *lower* dan *upper* saja.

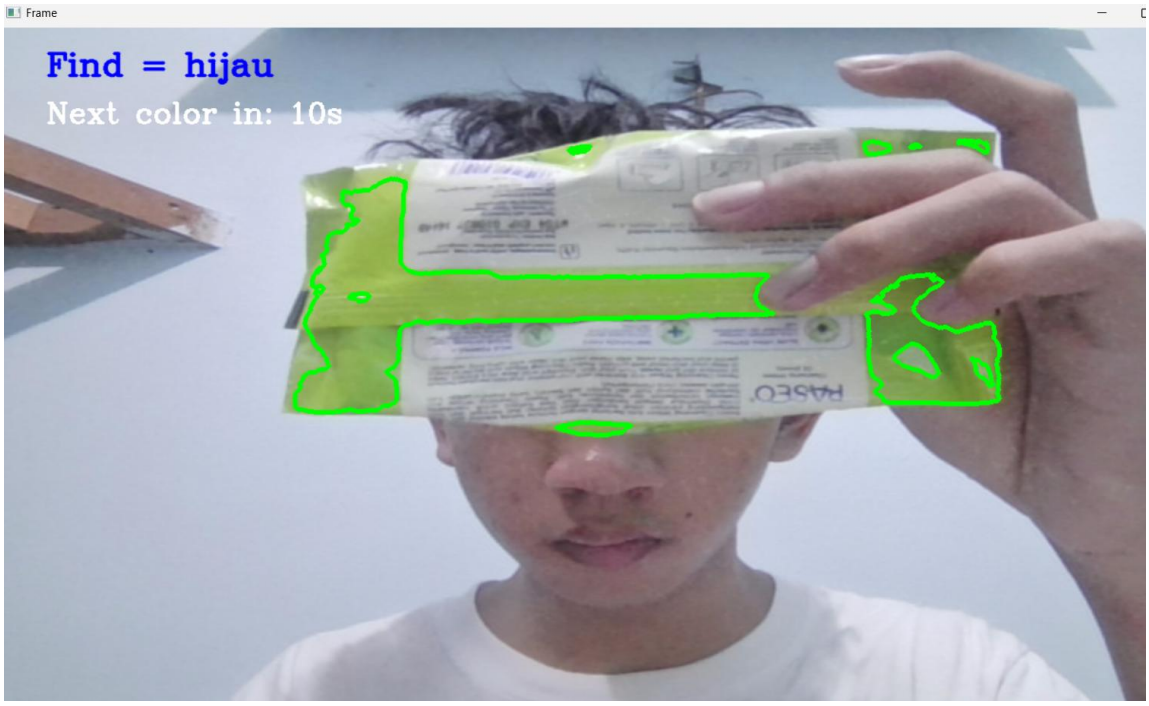
Kontur mulai dinisialisasi untuk mendeteksi ujung-ujung dari benda yang terdeteksi dalam gambar yang sudah di*masking*. Jika terdapat kontur, maka untuk setiap kontur yang ada akan dihitung luasnya dan jika luasnya lebih dari 1000 pixel maka kita akan menggambarkan kontur yang ada. Hal ini diharapkan dapat memasktikan jika hanya objek-objek besar saja yang terdeteksi bukan noise. Setelanya, variabel **`found`** akan berubah nilainya menjadi *True* untuk menyatakan benda dengan warna yang diinginkan berhasil terdeteksi. Variabel **`start_time`** juga akan diisi dengan nilai waktu tepat pada saat benda dengan warna tersebut terdeteksi.

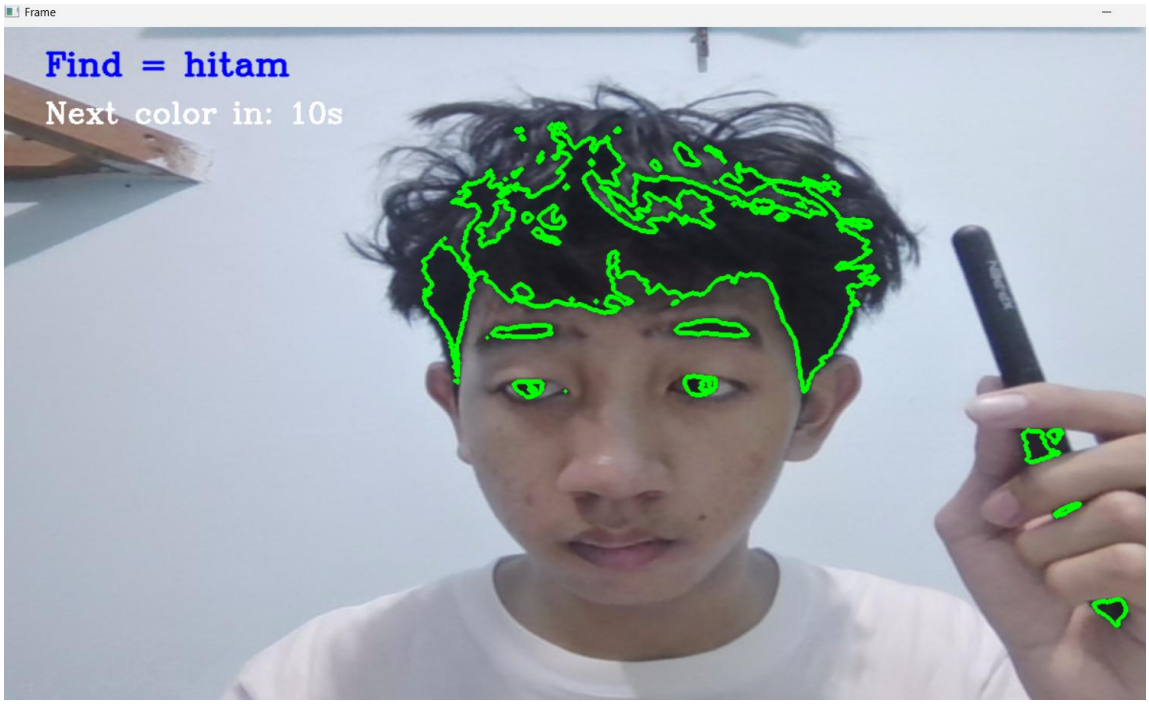
Selanjutnya, akan ada pengecekan jika benda berhasil ditemukan dan variabel **`start_time`** tidak bernilai *None*, maka program akan menghitung waktu yang sudah berjalan sejak benda itu terdeteksi dengan cara mengurangkan waktu saat ini dengan waktu saat pertama kali benda itu terdeteksi. Terdapat juga variabel **`remaining_time`** yang bertugas menyimpan waktu tersisa sebelum berganti warna, memiliki nilai maksimal 10 karena setelah 10 detik maka akan berganti warna seusai dengan kode

yang ditulis. Akan ada pengecekan apakah masih ada warna yang tersisa untuk dideteksi atau tidak. Jika tidak maka akan keluar dari program. Setelahnya program akan mengembalikan nilai variabel `found` dan `start_time` ke kondisi semula.

Setelahnya, program akan mengecek input keyboard pemain. Jika menekan “n” pada keyboard, program akan melewati warna ini dan berpindah ke warna lainnya. Program akan mengisi variabel `tidak_memiliki_warna` sebagai pengingat bahwa pemain tidak memiliki benda dengan warna terkait. Disamping itu jika pemain menekan “q” maka akan keluar dari permainan dan program akan dihentikan. Setelah keluar dari loop, terdapat dua baris kode akhir yang membersihkan memori.

3. Demonstrasi Kode





4. Saran

Dalam proyek ini, terdapat beberapa aspek yang masih perlu ditingkatkan dan ditinjau kembali guna meningkatkan akurasi serta keandalan sistem.

Pertama, terdapat ketidakkonsistenan dalam deteksi warna objek, yang dapat diperbaiki dengan penyesuaian nilai HSV yang lebih akurat. Saat ini, nilai HSV yang digunakan dalam proyek masih bersifat perkiraan dan belum dikalibrasi secara optimal, sehingga dapat menyebabkan deteksi warna yang kurang presisi. Oleh karena itu, diperlukan pendekatan yang lebih sistematis, seperti kalibrasi HSV berdasarkan sampel warna dari berbagai kondisi pencahayaan, atau bahkan menggunakan algoritma adaptif untuk menyesuaikan nilai HSV secara dinamis.

Kedua, dalam proses pendeteksian objek, terkadang program mendeteksi noise atau area tertentu dalam frame sebagai warna target, meskipun sebenarnya belum ada objek yang ditampilkan di depan kamera. Hal ini dapat mengakibatkan hasil yang keliru dan mengurangi akurasi sistem. Untuk mengatasi masalah ini, beberapa teknik dapat diterapkan, seperti penggunaan *edge detection* yang lebih baik, *thresholding* yang lebih presisi, serta kombinasi dengan metode deteksi objek lain, seperti *Haar Cascade* atau model yang telah dilatih sebelumnya (*pre-trained model*). Dengan pendekatan ini, sistem dapat lebih efektif dalam membedakan objek yang sebenarnya dengan gangguan atau noise di latar belakang, sehingga meningkatkan keakuratan deteksi warna.

Dengan melakukan perbaikan pada aspek-aspek tersebut, diharapkan proyek ini dapat menghasilkan deteksi warna yang lebih stabil, akurat, dan lebih andal dalam berbagai kondisi pencahayaan serta lingkungan yang berbeda.