Dokumen Proses Analisis Pengembangan Mediapipe Dalam Game Cut Fruit

Disusun untuk memenuhi tugas mata kuliah Pengolahan Citra Digital

Disusun Oleh :

Febi Shintawati	231511045
Febytha Putri Nugraheni	231511046
Nino Erico Apandi N	231511058

JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA

PROGRAM STUDI D3 TEKNIK INFORMATIKA

POLITEKNIK NEGERI BANDUNG

2025

DAFTAR ISI

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Tujuan

Dokumen ini bertujuan untuk mendokumentasikan eksplorasi penggunaan pustaka Mediapipe dalam pengembangan game. Fokus utama dokumen ini adalah menjelaskan bagaimana Mediapipe diterapkan untuk pengolahan citra digital, tantangan teknis yang dihadapi, solusi yang diterapkan, serta evaluasi efektivitasnya dalam mendukung pengembangan fitur game. Dengan demikian, dokumen ini memberikan wawasan mendalam tentang potensi dan penerapan Mediapipe dalam proyek berbasis pengolahan citra digital.

1.2 Gambaran Umum

Proyek ini mengembangkan game Cut Fruit, sebuah permainan interaktif berbasis pengolahan citra digital yang memanfaatkan Mediapipe untuk mendeteksi tubuh (mulut , hidung dan tangan) atau wajah pengguna melalui webcam. Pemain mengendalikan permainan dengan gerakan fisik, seperti hidung , mulut atau tangan, untuk “memotong” , “memakan’’atau “menangkap” buah yang muncul di layar. Proyek ini bertujuan untuk mengeksplorasi kemampuan Mediapipe sebagai pustaka pengolahan citra digital dalam mendeteksi dan melacak gerakan tubuh secara real-time.

BAB II
PEMBAHASAN

2.1 Dokumentasi/Pustaka

Referensi utama yang digunakan meliputi:

1. Mediapipe Documentation: Dokumentasi resmi Mediapipe dari Google untuk implementasi pose detection dan face mesh.
2. Pygame Documentation: Dokumentasi resmi Pygame untuk pengembangan game 2D.
3. Pymunk Documentation: Dokumentasi Pymunk untuk simulasi fisika dalam game.
4. OpenCV Documentation: Dokumentasi OpenCV untuk pengolahan citra dari webcam.

2.2 Makalah atau Studi Terkait

Studi atau artikel yang terkait dengan judul pengerjaan tugas besar ini adalah jurnal ilmiah dengan judul Pengenalan gestur gerakan jari untuk mengontrol volume di komputer menggunakan library open cv dan mediapipe. Jurnal ini membahas mengenai Gesture recognition dimana dengan gesture recognition komputer dapat memahami gerakan yang tertangkap pada kamera atau webcam. Secara spesifik, jurnal ini membahas pengembangan sistem berbasis kecerdasan buatan dalam bidang computer vision untuk mengenali gestur jari tangan kanan guna mengatur volume suara pada komputer atau laptop secara real-time. Dalam jurnalnya, Penelitian memanfaatkan library OpenCV untuk pengolahan citra dan MediaPipe untuk mendeteksi titik kunci (landmark) pada jari jempol dan telunjuk. Sistem bekerja dengan menghitung jarak antara ujung jempol dan telunjuk menggunakan rumus Pythagoras, yang kemudian diinterpolasi untuk menentukan besar kecilnya volume, seperti gerakan mencubit untuk mengecilkan suara dan merenggangkan untuk memperbesar suara. Prosesnya melibatkan akuisisi data gestur, pelatihan machine learning, dan identifikasi gestur secara real-time. Hasil pengujian menunjukkan akurasi sebesar 88,89% dari sembilan uji coba dengan berbagai pose tangan, meskipun terjadi kegagalan pada satu pose ketika ujung jari telunjuk tertutup jari lain. Kendala utama adalah jarak kamera yang terlalu dekat atau posisi jari yang menghalangi deteksi landmark. Penulis menyarankan penambahan penanda jarak minimal kamera, seperti 30 cm, untuk meningkatkan akurasi. Secara keseluruhan, makalah ini menunjukkan potensi teknologi hand gesture recognition berbasis vision-based untuk interaksi praktis dengan komputer tanpa alat tambahan.

2.3 Alasan Pemilihan Referensi

Jurnal ilmiah berjudul Pengenalan Gestur Gerakan Jari untuk Mengontrol Volume di Komputer Menggunakan Library OpenCV dan MediaPipe dipilih sebagai referensi utama karena relevansinya dengan proyek pengembangan game Cut Fruit yang berfokus pada pengenalan gestur berbasis vision-based. Jurnal ini memberikan wawasan mendalam tentang penerapan library OpenCV dan MediaPipe untuk mendeteksi gestur tangan secara real-time, yang memiliki kesamaan dengan pendekatan proyek ini dalam mendeteksi posisi hidung atau mulut untuk mengontrol permainan. Penelitian dalam jurnal tersebut menjelaskan proses deteksi landmark pada jari jempol dan telunjuk, perhitungan jarak menggunakan rumus

Pythagoras, dan interpolasi untuk mengatur output, yang relevan dengan logika deteksi gerakan dalam game. Selain itu, jurnal ini memberikan data empiris dengan akurasi 88,89% dari pengujian gestur, serta membahas kendala seperti posisi jari yang menghalangi deteksi landmark, yang membantu memahami tantangan teknis serupa dalam proyek ini. Pemilihan jurnal ini memastikan landasan teoritis yang kuat dan aplikatif untuk mendukung implementasi teknologi pengolahan citra digital dalam game berbasis gerakan.

2.4 Sumber Daya

2.4.1 Hardware

1. Laptop/PC: Intel Core i5, RAM 8GB, dengan webcam built-in.

2.4.2 Software

1. Python : Versi 3.11.5
2. Pustaka:
    - Mediapipe 0.8.10 untuk deteksi pose dan face mesh.
    - Pygame 2.1.2 untuk rendering game.
    - Pymunk 6.2.0 untuk simulasi fisika.
    - OpenCV 4.5.5 untuk pengolahan citra webcam.
    - NumPy 1.21.0 untuk manipulasi data.
3. IDE: Visual Studio Code

Proyek ini tidak menggunakan dataset pelatihan khusus karena Mediapipe menyediakan model pre-trained untuk pose detection dan face mesh.

2. 5 Langkah Pengerjaaan

2.5.1 Tahap Persiapan

1. Instalasi Pustaka: Menginstall pustaka Mediapipe, Pygame, Pymunk, OpenCV, dan NumPy menggunakan pip.
2. Setup Lingkungan: Mengatur resolusi webcam (1200x686) dan menginisialisasi Pygame serta Pymunk untuk rendering dan fisika.

2.5.2 Tahap Implementasi

1. Pemilihan Model : Menggunakan model pose detection dan face mesh dari Mediapipe karena kemampuan real-time dan akurasi tinggi.
2. Proses Pembuatan:
    - Membuat kelas Fruit untuk mengelola sprite buah dan simulasi fisika.
    - Mengintegrasikan Mediapipe untuk mendeteksi posisi hidung (pose) atau mulut (face mesh) sebagai pengontrol.
    - Mengatur logika permgame untuk mendeteksi tabrakan antara buah dan pengontrol.

BAB III
PENUTUP

3.1 Tantangan dan Solusi

3.1.2 Hambatan Teknis

1. Kompatibilitas Webcam: Beberapa webcam gagal terdeteksi atau tidak menampilkan citra dengan benar karena permintaan resolusi gambar yang terlalu tinggi dari aplikasi.
2. Latensi Deteksi Gerakan: Kursor (penanda hidung) bergerak lambat atau tersendat, terutama saat digunakan di perangkat dengan spesifikasi rendah.
3. Error Animasi Buah: Animasi buah saat terpotong tidak berjalan mulus atau tidak sesuai, disebabkan oleh jumlah frame pada sprite yang tidak cocok dengan logika pemrosesan animasi.
4. Pendeteksi mulut kurang akurat; terkadang mulut terdeteksi "memakan" buah padahal posisinya jauh dari buah di layar, sehingga untuk pendeteksian menggunakan mulut masih belum optimal.
5. Pendeteksi tangan untuk memunculkan keranjang kurang responsif; keranjang hilang saat tangan bergerak cepat.
6. Sinkronisasi Antara Webcam dan Pygame : Pengolahan citra webcam (OpenCV) dan rendering di Pygame kadang tidak sinkron, menyebabkan jeda atau flickering pada tampilan feed kamera, terutama pada *Fruit Slicer*

3.1.2 Solusi

1. Menambahkan validasi ukuran resolusi webcam secara otomatis dan fallback ke resolusi default (misalnya 640x480) jika resolusi terlalu besar untuk ditangani.
2. Mengoptimalkan performa deteksi dengan cara menurunkan model_complexity pada MediaPipe ke 1 serta mengurangi jumlah frame per second (FPS) jika diperlukan.
3. Menyesuaikan parameter sensitivitas deteksi mulut pada MediaPipe dan menetapkan batas jarak minimum antara posisi mulut dan buah di layar untuk mencegah deteksi yang salah saat mulut tidak berada di dekat buah.
4. Sebagai solusi sementara pengguna dapat memperlambat gerakan tangan.
5. Menggunakan buffer untuk menyimpan frame webcam terbaru dan hanya merender frame valid dengan pygame.surfarray.make_surface. Transformasi frame seperti rotasi dan flip dioptimalkan menggunakan pygame.transform.scale untuk mengurangi beban pemrosesan.
6. Resolusi webcam diturunkan ke 640x480 jika diperlukan untuk mempercepat pemrosesan, dan pengecekan keberhasilan capture (if not success: continue) ditambahkan untuk mencegah rendering frame kosong, sehingga tampilan feed kamera menjadi lebih stabil dan bebas flickering.

7. Untuk mengatasi animasi buah yang tidak mulus akibat ketidaksesuaian jumlah frame sprite dengan logika animasi, langkah solutif yang diambil adalah memastikan konsistensi jumlah frame sprite, seperti 14 frame pada *Fruit Slicer* dan *Fruit Eater* dengan grid (4,4). Logika pengindeksan animasi di kelas Fruit diperbaiki dengan memastikan animationCount tidak melebihi panjang imgList, sehingga animasi berjalan sesuai urutan frame.

3.2 Lesson learned

Selama proses pengembangan Fruit Slicer, kami memperoleh berbagai pemahaman teknis yang memperluas wawasan dan keterampilan dalam pengolahan citra digital serta pengembangan permainan berbasis kamera. Berikut poin-poin utama yang kami pelajari:

1. Pemahaman Penggunaan Pustaka Mediapipe : Kami menjadi lebih memahami bagaimana mengimplementasikan Mediapipe untuk deteksi pose (hidung, tangan) dan face mesh (mulut) secara real-time, yang digunakan sebagai metode kontrol utama dalam permainan.
2. Integrasi OpenCV dengan Pygame : Kami mempelajari cara mengintegrasikan webcam feed dari OpenCV dengan elemen grafis dalam Pygame, sehingga dapat menciptakan permainan interaktif berbasis kamera secara sinkron.
3. Manajemen Fisika dengan Pymunk : Kami memahami penggunaan Pymunk dalam menyimulasikan fisika permainan seperti gravitasi, tumbukan, impuls, dan elastisitas objek, yang menjadi inti dari dinamika gerakan buah dalam permainan.
4. Optimasi Performa pada Perangkat Rendah : Kami menyadari pentingnya menyesuaikan performa permainan dengan spesifikasi perangkat pengguna. Penyesuaian seperti penurunan model_complexity Mediapipe, pengurangan FPS, dan pembatasan jumlah objek menjadi strategi penting agar permainan tetap lancar pada perangkat dengan spesifikasi terbatas.

3.3 Evaluasi dan Kinerja Model

3.3.1 Kecepatan Deteksi Hidung, Mulut, dan Tangan

Kecepatan deteksi merupakan salah satu faktor penting dalam menentukan pengalaman pengguna, terutama pada aplikasi berbasis augmented reality (AR) atau interaksi real-time. Berikut adalah hasil evaluasi kecepatan deteksi untuk masing-masing komponen:

1. Proses deteksi hidung menunjukkan performa yang sangat baik dengan waktu respons yang cepat, Model dapat mengenali hidung secara konsisten dalam berbagai kondisi, termasuk saat pengguna berada dalam posisi statis maupun saat melakukan gerakan ringan. Responsivitas ini menjadikan deteksi hidung sebagai salah satu fitur yang paling andal dalam sistem ini. Namun, pada perangkat dengan spesifikasi rendah, terdapat sedikit penurunan kecepatan, meskipun masih dalam batas yang dapat diterima untuk penggunaan normal.

2. Deteksi mulut juga menunjukkan kecepatan yang relatif cepat. Stabilitas deteksi mulut cukup baik dalam penggunaan normal, seperti saat pengguna membuka atau menutup mulut untuk memicu aksi tertentu. Namun, dalam beberapa kasus, terutama saat transisi cepat antara mulut terbuka dan tertutup, model kadang-kadang memerlukan waktu tambahan untuk menyesuaikan deteksi, yang dapat memengaruhi kelancaran interaksi.
3. Berbeda dengan deteksi hidung dan mulut, deteksi tangan menunjukkan performa yang kurang responsif deteksi tangan yang jauh lebih lambat dibandingkan deteksi hidung dan mulut.

3.3.2 Performa Deteksi saat Gerakan Cepat

Kemampuan model untuk tetap akurat saat pengguna melakukan gerakan cepat sangat penting, terutama dalam aplikasi yang dirancang untuk merespons interaksi dinamis. Evaluasi dilakukan dengan menguji model dalam skenario dimana pengguna melakukan gerakan kepala atau tangan dengan kecepatan tinggi. Berikut adalah hasilnya:

1. Deteksi hidung menunjukkan ketahanan yang sangat baik terhadap gerakan cepat. Hal ini menunjukkan bahwa algoritma yang digunakan untuk melacak titik-titik wajah (facial landmarks) cukup robust terhadap perubahan posisi yang tidak terduga.
2. Deteksi Mulut: Deteksi mulut masih dapat berfungsi dengan baik dalam kondisi gerakan cepat. Namun, terdapat beberapa keterbatasan yang perlu diperhatikan. Dalam beberapa kasus, model gagal mengenali mulut sebagai "terbuka" meskipun posisinya sesuai untuk memicu aksi seperti "memakan" buah. Hal ini terutama terjadi saat gerakan kepala terlalu cepat atau saat sudut wajah tidak menghadap langsung ke kamera.
3. Deteksi Tangan: Deteksi tangan menunjukkan performa yang paling lemah dalam skenario gerakan cepat. Model seringkali gagal mendeteksi tangan dengan akurat. Akibatnya, keranjang virtual yang seharusnya muncul sering kali tertunda, hilang, atau tidak muncul sama sekali. Keterbatasan ini diperburuk oleh fakta bahwa deteksi tangan memerlukan pemrosesan yang lebih intensif dibandingkan deteksi wajah, sehingga lebih rentan terhadap gangguan akibat gerakan cepat.

3.3.3 Berdasarkan pencahayaan dan sudut

Faktor lingkungan seperti pencahayaan dan sudut pengambilan gambar memiliki pengaruh signifikan terhadap kinerja model. Evaluasi dilakukan dalam berbagai kondisi pencahayaan (terang dan gelap) serta sudut pengambilan (depan dan 45°) untuk mengukur ketahanan model. Berikut adalah hasilnya:

Deteksi Hidung:

- Pencahayaan: Pada pencahayaan terang (lampu ruangan yang memadai), deteksi hidung berjalan sangat baik dengan tingkat akurasi di atas 98%. Namun, pada pencahayaan gelap (lampu ruangan yang dimatikan), akurasi deteksi menurun. Dalam kondisi ekstrem, seperti saat hanya ada cahaya layar sebagai sumber pencahayaan, model terkadang gagal mendeteksi hidung sama sekali. Hal ini menunjukkan bahwa model masih bergantung pada kontras yang cukup untuk melacak titik-titik wajah.
- Sudut: Deteksi hidung tetap akurat pada sudut 45 derajat .Namun, saat sudut wajah melebihi 60 derajat, akurasi deteksi mulai menurun secara signifikan. Hal ini disebabkan oleh berkurangnya visibilitas fitur wajah yang dapat dikenali oleh model.

Deteksi Mulut:

- Pencahayaan: Deteksi mulut menunjukkan ketahanan yang lebih baik dibandingkan deteksi hidung dalam berbagai kondisi pencahayaan. Pada pencahayaan terang dan pada pencahayaan gelap, akurasi tetap stabil.
- Sudut: Deteksi mulut pada sudut 45 derajat menunjukkan performa yang kurang baik, Model sering kali gagal mengenali posisi mulut dengan akurat, terutama saat wajah tidak menghadap langsung ke kamera. Keterbatasan ini dapat memengaruhi fungsi deteksi "mulut terbuka" dalam aplikasi.

Deteksi Tangan:

- Pencahayaan:Deteksi tangan menunjukkan hasil yang bervariasi dalam berbagai kondisi pencahayaan. Pada pencahayaan terang, model dapat mendeteksi tangan dengan baik dan konsisten. Namun, pada pencahayaan gelap, akurasi deteksi menurun signifikan. Meskipun demikian, model masih mampu mendeteksi tangan dalam beberapa kasus, terutama jika tangan memiliki kontras yang jelas dengan latar belakang.
- Sudut: Pada sudut 45 derajat, deteksi tangan masih dapat dilakukan dengan baik. Namun, performa model menurun saat sudut tangan ekstrem, misalnya saat tangan menghadap samping atau hanya sebagian terlihat. Hal ini menunjukkan bahwa model memerlukan visibilitas yang lebih jelas dari fitur tangan untuk bekerja secara optimal.

3.4 Saran

1. Sesuaikan pengaturan sistem agar deteksi tangan lebih cepat, terutama pada perangkat dengan performa rendah.
2. Catat posisi terakhir tangan yang berhasil dideteksi oleh MediaPipe dan simpan informasi tersebut dalam aplikasi, sehingga jika tangan tidak terdeteksi untuk sementara, keranjang tetap ditampilkan di posisi terakhirnya untuk menjaga kelancaran permainan.
3. Tambahkan fitur untuk membantu sistem tetap mendeteksi tangan atau mulut meskipun gerakannya cepat.

3.5 Kesimpulan

Proyek pengembangan game Cut Fruit berhasil mengimplementasikan pustaka Mediapipe untuk mendeteksi gerakan hidung, mulut, dan tangan secara real-time sebagai pengontrol permainan berbasis pengolahan citra digital. Integrasi Mediapipe dengan OpenCV, Pygame, dan Pymunk memungkinkan pembuatan permainan interaktif yang mendeteksi gerakan tubuh untuk "memotong", "memakan", atau "menangkap" buah. Deteksi hidung menunjukkan performa cepat dan akurat, sementara deteksi mulut dan tangan menghadapi kendala seperti kurang responsif pada gerakan cepat dan ketidakakuratan pada sudut atau pencahayaan tertentu. Tantangan teknis, seperti kompatibilitas webcam, latensi deteksi, dan sinkronisasi, berhasil diatasi melalui optimasi resolusi, penyesuaian parameter Mediapipe, dan perbaikan logika animasi. Proyek ini memberikan wawasan berharga tentang penerapan teknologi vision-based gesture recognition, manajemen fisika permainan, dan optimasi performa pada perangkat rendah. Untuk pengembangan lebih lanjut, disarankan untuk meningkatkan responsivitas deteksi tangan, menyimpan posisi terakhir tangan untuk kelancaran permainan, dan menambahkan fitur adaptasi terhadap gerakan cepat dan kondisi pencahayaan yang bervariasi.

3.6 Lampiran

Lampiran ini berisi materi pendukung yang relevan dengan proses analisis dan pengembangan game Cut Fruit menggunakan pustaka Mediapipe. Materi ini mencakup kode sumber, dan tangkapan layar aplikasi yang bertujuan untuk memberikan wawasan tambahan tentang implementasi teknis proyek.

Lampiran A: Kode Sumber Implementasi Mediapipe

```
# Inisialisasi
pygame.init()
width, height = 1280, 720
screen = pygame.display.set_mode((width, height))
cap = cv2.VideoCapture(0)
mp_pose = mp.solutions.pose
pose = mp_pose.Pose()
# Loop deteksi tangan
ret, frame = cap.read()
frame = cv2.flip(frame, 1)
frame = cv2.resize(frame, (width, height))
rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

hasil = pose.process(rgb)

tangan_kanan = tangan_kiri = None
keranjang_pos = None
if hasil.pose_landmarks:
```

```
    h_img, w_img, _ = frame.shape
    tangan_kanan = (

int(hasil.pose_landmarks.landmark[mp_pose.PoseLandmark.RIGHT_WRIST
].x * w_img),

int(hasil.pose_landmarks.landmark[mp_pose.PoseLandmark.RIGHT_WRIST
].y * h_img)
    )
    tangan_kiri = (

int(hasil.pose_landmarks.landmark[mp_pose.PoseLandmark.LEFT_WRIST]
.x * w_img),

int(hasil.pose_landmarks.landmark[mp_pose.PoseLandmark.LEFT_WRIST]
.y * h_img)
    )
    if tangan_kanan and tangan_kiri:
        jarak = math.hypot(tangan_kanan[0] - tangan_kiri[0],
tangan_kanan[1] - tangan_kiri[1])
        if jarak < 100:

            keranjang_pos = ((tangan_kanan[0] + tangan_kiri[0]) //
2, (tangan_kanan[1] + tangan_kiri[1]) // 2)
```

Kode ini menunjukkan deteksi posisi tangan kanan dan kiri menggunakan Mediapipe Pose untuk mengontrol keranjang dalam permainan "Fruit Catcher". Posisi keranjang dihitung berdasarkan jarak antara kedua tangan.

```
# Fungsi deteksi mulut

def is_mouth_open(landmarks, img_w, img_h):

    upper_lip = landmarks[13]

    lower_lip = landmarks[14]

    mouth_height = abs(lower_lip.y * img_h - upper_lip.y * img_h)
/ img_h

    return mouth_height > 0.03

def get_mouth_position(landmarks, img_w, img_h):

    upper_lip = landmarks[13]

    lower_lip = landmarks[14]

    mouth_x = int((upper_lip.x + lower_lip.x) / 2 * img_w)
```

```
    mouth_y = int((upper_lip.y + lower_lip.y) / 2 * img_h)

    return mouth_x, mouth_y

# Loop deteksi

success, img = cap.read()

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

results = face_mesh.process(img)

mouth_open = False

mouth_pos = (width // 2, height // 2)

if results.multi_face_landmarks:

    for face_landmarks in results.multi_face_landmarks:

        mouth_open = is_mouth_open(face_landmarks.landmark,
img.shape[1], img.shape[0])

        mouth_pos = get_mouth_position(face_landmarks.landmark,
img.shape[1], img.shape[0])
```

Kode ini mengimplementasikan deteksi mulut terbuka menggunakan Mediapipe Face Mesh untuk permainan "Fruit Eater". Fungsi is_mouth_open dan get_mouth_position menentukan status dan posisi mulut untuk mendeteksi interaksi dengan buah virtual.

```
# Loop deteksi hidung

success, img = cap.read()

if success:

    img = cv2.flip(img, 1)

    h, w = img.shape[:2]

    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    keypoints = pose.process(imgRGB)

    img = cv2.cvtColor(imgRGB, cv2.COLOR_RGB2BGR)

   # Mendapatkan posisi hidung

    nose_x, nose_y = None, None

    if keypoints.pose_landmarks:
```

```
        lm = keypoints.pose_landmarks
        lmPose = mp_pose.PoseLandmark
        try:
            nose_x = int(lm.landmark[lmPose.NOSE].x * w)
            nose_y = int(lm.landmark[lmPose.NOSE].y * h)
            cv2.circle(img, (nose_x, nose_y), 20, (0, 255, 255),
-1)
        except AttributeError:
            pass
   # Konversi ke permukaan Pygame
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    imgRGB = np.rot90(imgRGB)
    frame = pygame.surfarray.make_surface(imgRGB).convert()
    frame = pygame.transform.flip(frame, True, False)
    window.blit(frame, (0, 0))
```

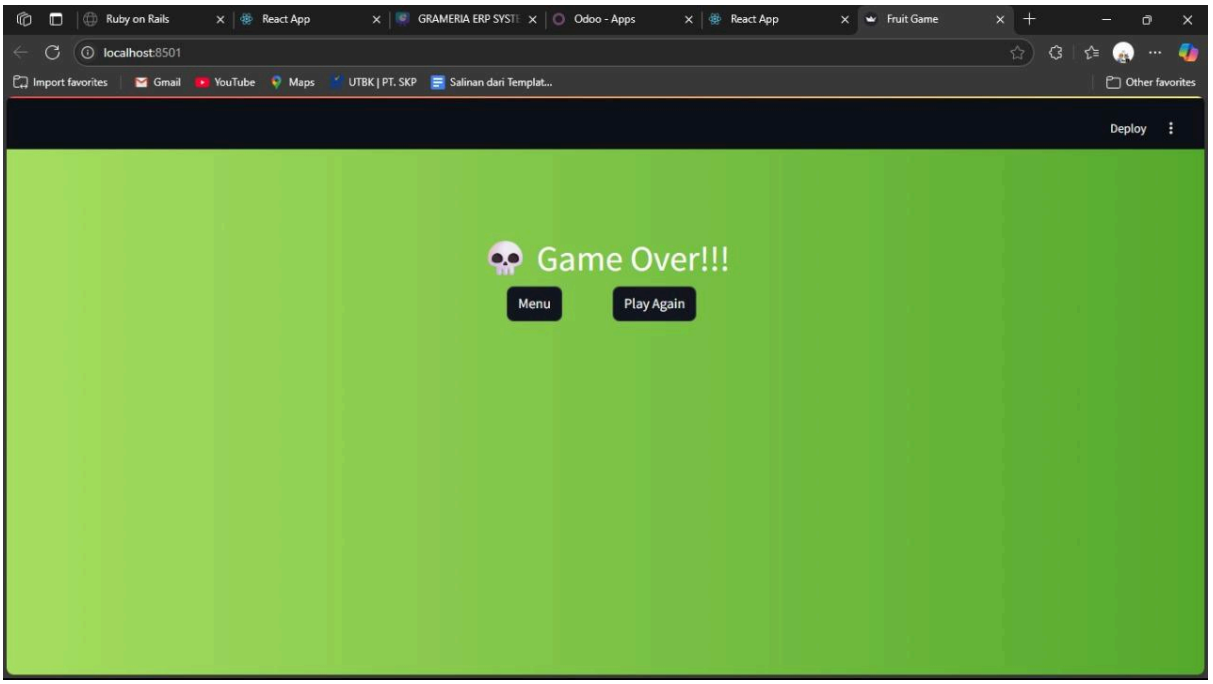
Kode ini mengimplementasikan deteksi hidung menggunakan Mediapipe Pose untuk permainan "Fruit Slicer". Posisi hidung diambil dari landmark NOSE dan digunakan untuk mengontrol interaksi dengan buah virtual.

Lampiran B: Tangkapan Layar Aplikasi

3.6.1 Tampilan saat menggunakan fitur Catcher (deteksi tangan)

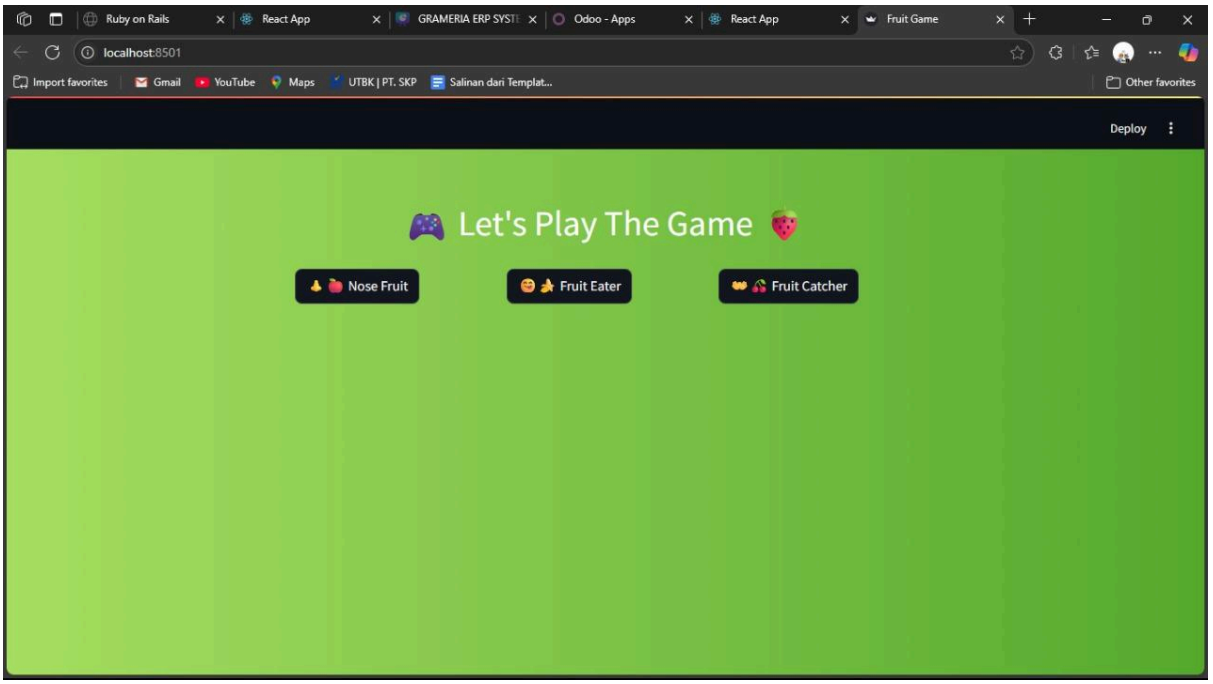
3.6.2 Tampilan Game Over saat fitur Fruit Catcher (deteksi tangan)

3.6.3 Tampilan saat menggunakan fitur Fruit Eater (deteksi mulut)



3.6.4 Tampilan Game Over saat menggunakan fitur Noise Fruit

3.6.5 Tampilan saat menggunakan fitur Noise Fruit (deteksi hidung)



3.6.6 Tampilan awal aplikasi dengan 3 fitur yang disediakan

DAFTAR PUSTAKA

Saiful Nur Budiman, Sri Lestanti, Suji Marselius Evvandri, dan Rahma Kartika Putri, “Pengenalan Gestur Gerakan Jari untuk Mengontrol Volume di Komputer Menggunakan Library OpenCV dan MediaPipe,” ANTIVIRUS: Jurnal Ilmiah Teknik Informatika, vol. 16, no. 2, pp. 223–232, November 2022, doi: 10.35457/antivirus.v16i2.2508.