DOKUMEN ANALISIS PROSES PENGEMBANGAN GAME FLAPPY BIRD BERBASIS KONTROL WAJAH MENGGUNAKAN FACE DETECTION DENGAN MEDIAPIPE FACE MESH

Disusun untuk memenuhi tugas besar mata kuliah Pengolahan Citra Digital (Praktek)

Disusun oleh:

Farhan Maulana	*231511040*
Indah Ratu Pramudita	*231511050*
Nazla Kayla	*231511057*

JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA
D3 TEKNIK INFORMATIKA
POLITEKNIK NEGERI BANDUNG
2025

DAFTAR ISI

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1. Latar Belakang

Kemajuan teknologi computer vision telah memungkinkan interaksi antara manusia dan komputer menjadi lebih natural dan intuitif. Salah satu pendekatan yang semakin banyak digunakan adalah pemanfaatan model *pre-trained* untuk mendeteksi dan melacak bagian tubuh manusia melalui kamera secara real-time. MediaPipe Face Mesh adalah salah satu teknologi unggulan dari Google yang mampu mendeteksi titik-titik wajah secara detail, termasuk hidung, mata, dan bibir, dengan kecepatan dan akurasi tinggi.

Sementara itu, game *Flappy Bird* yang sempat viral karena mekanismenya yang sederhana namun menantang, membuka peluang eksplorasi di bidang interaksi berbasis wajah. Dengan menggabungkan teknologi MediaPipe dan konsep game klasik ini, sebuah inovasi interaktif dapat diwujudkan, di mana pengguna tidak lagi menggunakan *keyboard* atau *mouse*, melainkan cukup dengan menggerakkan hidung di depan kamera untuk mengontrol objek dalam permainan.

Implementasi sistem ini tidak hanya menjadi sarana pembelajaran teknologi *computer vision*, tetapi juga melibatkan pemahaman terhadap integrasi sistem web modern. Sistem dibangun menggunakan FastAPI sebagai *backend* dan Tailwind CSS untuk *styling* antarmuka, serta memanfaatkan WebSocket untuk komunikasi data secara real-time. Proyek ini juga sepenuhnya *offline* dan *single player*, memastikan bahwa permainan dapat dijalankan tanpa ketergantungan pada koneksi internet.

1.2. Tujuan

Adapun tujuan dari proyek tugas besar ini adalah:

1. Mengimplementasikan deteksi posisi hidung menggunakan MediaPipe Face Mesh secara *real-time*.
2. Mengintegrasikan hasil deteksi ke dalam game Flappy Bird sebagai kontrol utama pergerakan burung.
3. Membangun sistem permainan berbasis *web* yang responsif dan dapat dijalankan secara lokal tanpa koneksi internet.

4. Menerapkan FastAPI sebagai *backend* server dan Tailwind CSS untuk antarmuka pengguna.
5. Melakukan eksplorasi berupa evaluasi dari berbagai pengujian untuk menguji sejauh mana akurasi dari model *pre-trained* MediaPipe, khususnya posisi hidung.

1.3. Manfaat

Manfaat yang dapat diperoleh dari implementasi tugas besar ini antara lain:

1. Memberikan pengalaman langsung dalam membangun aplikasi computer vision berbasis *pre-trained model*.
2. Menunjukkan penerapan konsep interaksi manusia-komputer melalui kontrol berbasis wajah.
3. Menjadi dasar pengembangan aplikasi hiburan atau terapi interaktif dengan kontrol berbasis gerakan wajah.
4. Melatih kemampuan dalam membangun sistem terintegrasi mulai dari *backend*, *frontend*, hingga komunikasi *real-time*.
5. Menunjukkan kemampuan penggunaan MediaPipe secara kreatif dan spesifik.

1.4. Ruang Lingkup

Ruang lingkup proyek ini dibatasi pada:

1. Implementasi deteksi posisi hidung menggunakan model MediaPipe Face Mesh yang sudah tersedia (pre-trained).
2. Pengembangan game Flappy Bird versi web sederhana berbasis HTML, CSS (TailwindCSS), dan JavaScript, yang dihubungkan dengan *backend* FastAPI.
3. Sistem hanya digunakan secara lokal (tanpa koneksi internet) dan untuk satu pemain.
4. Skor dan highscore disimpan dalam file teks lokal.
5. Pengujian dilakukan dalam kondisi pencahayaan dan sudut pandang kamera yang umum digunakan pada perangkat laptop atau PC.

BAB II
TINJAUAN PUSTAKA

2.1. Flappy Bird

Flappy Bird adalah permainan sederhana yang tersedia di platform Android dan iOS, diciptakan oleh Nguyen Ha Dong, seorang pengembang asal Hanoi, Vietnam, dan dirilis pada Mei 2013. Permainan ini dimainkan dengan cara mengetuk layer ponsel untuk membuat burung terbang melewati pipa-pipa hijau. Jika waktu ketukan tidak tepat, burung bisa menabrak pipa, sehingga pemain harus mengulang permainan dari awal. Nguyen menyebutkan bahwa permainan ini dirancang untuk memberikan pengalaman santai bagi para pemainnya.

Permainan ini pada awalnya bernama Flap Flap dengan gaya visual yang terinspirasi dari permainan nitendo. Nguyen hanya membutuhkan waktu dua hari untuk menyelesaikan pembuatannya. Namun, karena nama Flap Flap sulit digunakan di App Store, Nguyen mengubahnya menjadi Flappy Bird pada 224 Mei 2013, dirilis melalui .Gear Studio, studio independen miliknya di Vietnam. Pada hari yang sama ia mengunggah cuitan di Twitter tentang skor tertingginya, sekaligus membagikan tautan untuk mengunduh permainan tersebut. Flappy Bird menempati peringkat 1.469 dalam kategori aplikasi ”Keluarga” di App Store untuk wilayah Amerika Serikat, menunjukkan bahwa permainan ini mulai dikenal sebagai salah satu permainan keluarga.

2.2. Mediapipe FaceMesh

Mediapipe Face Mesh adalah pengolahan visi komputer yang dikembangkan oleh Google untuk mendeteksi dan melacak fitur wajah secara *real-time* dengan akurasi tinggi. Teknologi ini memungkinkan ekstraksi hingga 468 titik koordinat (*landamarks*) pada wajah manusia yang mencakup detail seperti kontur wajah, mata, alis, hidung, dan mulut, dalam format tiga dimensi (3D) termasuk sumbu x, y, dan z, yang merepresentasikan posisi relatif fitur wajah terhadap kamera. Koordinat ini dapat digunakan untuk berbagai aplikasi, seperti pengenalan ekspresi wajah, pelacakan gerakan kepala, atau kontrol interaktif berbasis wajah. MediaPipe Face Mesh tersedia dalam berbagai platform, termasuk TensorFlow.js, yang memungkinkan implementasinya pada aplikasi berbasis web secara langsung di peramban tanpa memerlukan perangkat keras khusus (Subbiah, 2023).

MediaPipe Face Mesh diimplementasikan dalam TensorFlow.js memungkinkan pemrosesan langsung di sisi klien (client-side) melalui peramban web, memanfaatkan WebGL untuk akselerasi perangkat keras. Hal ini membuatnya sangat portabel dan dapat dijalankan pada perangkat dengan sumber daya terbatas, seperti ponsel atau komputer dengan spesifikasi standar, tanpa memerlukan server tambahan untuk pemrosesan. Berikut merupakan beberapa tahapan utama dari cara kerja Mediapipe Face Mesh:

1. Deteksi Wajah (*Face Detection*)

   Mendeteksi keberadaan wajah dalam bingkai (*frame*) video atau gambar menggunakan algoritma deteksi wajah.
2. Pemetaan *Landmarks*

   Setelah wajah terdeteksi, model memetakan 468 titik *landmarks* wajah dengan memprediksi koordinat 3D untuk setiap titik berdasarkan fitur wajah yang diidentifikasi, seperti sudut mata, ujung hidung, atau garis bibir.
3. Pelacakan *Real-time*

   Model ini dioptimalkan untuk melacak perubahan posisi wajah kontinu dalam aliran video, untuk memastikan responsivitas tinggi bahkan pada kecepatan bingkai (*fram rate*) yang tinggi
4. *Output* Data

   Hasil pemrosesan berupa array koordinat *landmarks* dalam format JSON atau struktur data serupa, yang dapat digunakan untuk analisis atau integrasi dengan aplikasi.

MediaPipe Face Mesh menggunakan model yang telah dilatih sebelumnya (*pre-trained model*) yang dioptimalkan untuk performa dan akurasi, memungkinkan pengembang untuk mengintegrasikannya dengan mudah tanpa perlu melatih ulang model. Dalam TensorFlow.js, model ini dijalankan menggunakan JavaScript, dengan dukungan untuk akselerasi GPU melalui WebGL, sehingga pemrosesan wajah dapat dilakukan secara efisien di peramban.

MediaPipe Face Mesh dapat mendeteksi gerakan wajah, seperti kedipan mata atau kemiringan kepala, yang dapat diterjemahkan menjadi perintah dalam game, misalnya membuat burung melompat saat pengguna mengedipkan mata. Koordinat landmarks wajah yang dihasilkan memberikan data presisi untuk memetakan gerakan wajah ke aksi permainan. Kecepatan pemrosesan MediaPipe Face Mesh memastikan bahwa gerakan wajah pengguna dapat diterjemahkan menjadi perintah permainan dengan latensi minimal, yang penting untuk menjaga pengalaman bermain yang responsif dan mulus. Data landmarks wajah yang dihasilkan oleh

MediaPipe Face Mesh dapat dikirim ke server melalui WebSocket untuk pemrosesan tambahan (misalnya, analisis gerakan wajah untuk skor atau mode multiplayer) atau diintegrasikan dengan API yang dibuat menggunakan FastAPI untuk menyimpan data permainan, seperti skor tertinggi.

2.3. WebSocket

WebSocket adalah protokol komunikasi dua arah yang berjalan di atas protokol TCP, memungkinkan pertukaran data secara *real-time* antara klien dan server dalam lingkungan yang terkontrol (Fette, 2011). Berbeda dengan protokol HTTP trasisional yang menggunakan model permintaan respons, WebSoccket memungkinkan koneksi tetap terbuka sehingga klien dan server dapat saling mengirim pesan secara sinkronus tanpa perlu memulai ulang koneksi. Protokol ini dirancang untuk mengurangi latensi dan *overhead* jaringan dibandingkan metode seperti long polling, seperti BOSH (Bidirectional-streams Over Synchronous HTTP), yang digunakan sebelumnya untuk komunikasi *real-time*.

Keunggulan utama WebSocket adalah kemampuannya untuk menyediakan komunikasi *full-duplex* sehingga cocok untuk aplikasi yang memerlukan pembaruan data secara langsung, seperti aplikasi perpesanan, notifikasi push, atau *game online*. Protokol ini menggunakan HTTP sebagai mekanisme transport awal untuk melakukan *handshake*, tetapi setelah koneksi terbentuk, komunikasi berlangsung secara asinkronus selama koneksi tetap terbuka. WebSocket juga mendukung standar RFC 6455 yang memastikan kompabilitas dan keamanan dalam pengiriman data, serta RFC7692 untuk ekstensi kompresi data, yang meningkatkan efisiensi pengiriman pesan.

Dalam konteks aplikasi *game online*, WebSocket menjadi pilihan yang relevan karena mendukung komunikasi *real-time* dengan antara komponen-komponen sistem. Game Flappy Bird berbasis kontrol wajah memerlukan pemrosesan data deteksi wajah secara *real-time* untuk menerjemahkan gerakan wajah (titik pada hidung) menjadi perintah dalam permainan, seperti membuat burung terbang. MediaPipe Face Mesh menghasilkan data koordinat wajah (*landmarks*) secara cepat, dan WebSocket dapat digunakan untuk mengirimkan data ini ke server atau modul pemrosesan lain dengan latensi rendah. WebSocket memungkinkan komunikasi dua arah yang asinkronus, sehingga data wajah dapat diproses dan diterjemahkan menjadi aksi permainan secara langsung tanpa penundaan signifikan

2.4. Fast API

FastAPI adalah kerangka kerja web modern berbasis Python yang dirancang untuk membangun API (*Application Programming Interface*) yang diluncurkan pada tahun 2018 oleh Sebastian Ramirez. FasAPI memanfaatkan fitur Python 3.7 seperti type hints dan pemrograman asinkronus untuk menyediakan pengembangan API yang cepat dan ramah pengembang. FastAPI secara otomatis menghasilkan dokumentasi API interaktif menggunakan standar OpenAPI (sebelumnya dikenal sebagai Swagger). Dokumentasi ini dapat diakses melalui endpoint seperti http://127.0.0.1:8000/docs (Swagger UI) atau http://127.0.0.1:8000/redoc (ReDoc), memungkinkan pengembang untuk menguji API langsung dari browser tanpa perlu menulis dokumentasi manual.

FastAPI menggunakan Pydantic untuk validasi dan serialisasi data. Dengan type hints Python, FastAPI dapat memvalidasi data masukan (seperti parameter jalur, query parameters, atau request body) secara otomatis, mengurangi risiko kesalahan data dan membuat kode lebih aman serta mudah dibaca. FastAPI memiliki sistem dependency injection yang memungkinkan pengembang untuk mendeklarasikan dependensi (seperti koneksi database atau autentikasi) secara modular. Ini meningkatkan kemudahan pengujian, pemeliharaan, dan skalabilitas kode

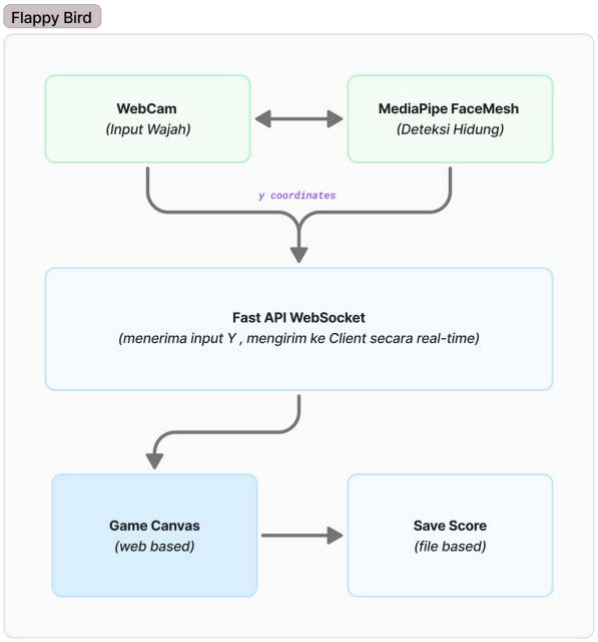
2.5. Tailwind CSS

Tailwind CSS adalah kerangka kerja (*framework*) CSS *open-source* yang dikembangkan oleh Adam Wathan, Jonathan Reinink, David Hemphill, dan Steve Schoger pada tahun 2017. Tailwind CSS menggunakan pendekatan *utility-first*, di mana pengembang dapat langsung menerapkan gaya visual ke elemen HTML menggunakan kelas-kelas utilitas yang telah ditentukan, seperti bg-blue-500 untuk warna latar belakang biru atau p-4 untuk padding 4 unit. Pendekatan ini memungkinkan fleksibilitas tinggi dalam desain tanpa meninggalkan struktur HTML, sehingga mempercepat proses pengembangan antarmuka pengguna (Flanoids, 2025).

Tailwind CSS dirancang untuk mendukung pengembangan aplikasi web yang responsif, modern, dan mudah disesuaikan. Dengan mengintegrasikan Tailwind CSS, pengembang dapat menciptakan desain yang konsisten dan estetis dengan kode yang lebih ringkas dibandingkan menulis CSS manual. Tailwind CSS juga mendukung konfigurasi kustom melalui file konfigurasi (tailwind.config.js), memungkinkan pengembang untuk menyesuaikan warna, ukuran, dan gaya sesuai kebutuhan proyek (Flanoids, 2025).

BAB III
IMPLEMENTASI DAN PEMBAHASAN

3.1. Desain Sistem



Gambar 1. Desain Sistem

Desain sistem pada proyek ini menggambarkan alur interaksi antara komponen utama yang membentuk aplikasi permainan Flappy Bird berbasis deteksi gerakan hidung menggunakan MediaPipe Face Mesh. Sistem dirancang agar dapat menangkap posisi hidung pengguna melalui kamera, mengirimkan data secara real-time ke permainan melalui WebSocket, dan menyimpan skor tertinggi secara lokal menggunakan backend FastAPI. Arsitektur ini memastikan game dapat berjalan secara responsif dan sepenuhnya *client-based* tanpa memerlukan koneksi internet eksternal.

3.2. Tahapan Pengembangan

Pengembangan sistem dilakukan secara bertahap untuk memastikan integrasi yang baik antara teknologi deteksi wajah, mekanisme permainan, serta komunikasi *real-time*. Setiap tahapan berfokus pada komponen utama yang saling terkait, dimulai dari konfigurasi awal hingga sistem siap digunakan secara penuh.

3.2.1. Persiapan Lingkungan

Tahap ini mencakup instalasi pustaka dan dependensi yang diperlukan untuk menjalankan sistem, seperti MediaPipe, FastAPI, dan Tailwind CSS. Selain itu, konfigurasi awal proyek juga mencakup pembuatan struktur folder, setup WebSocket, serta verifikasi perangkat keras seperti kamera untuk deteksi wajah secara real-time.

```
flappy_bird_web/
│
├── main.py                    # Server FastAPI
├── app/
│   ├── __init__.py
│   ├── game_logic.py          # Mesin permainan utama
│   ├── face_detection.py      # Deteksi wajah MediaPipe
│   └── websocket_handler.py   # Komunikasi WebSocket
│
├── static/
│   ├── index.html             # Antarmuka permainan utama
│   ├── game.js                # Logika permainan frontend
│   └── assets/                # Sprite dan gambar permainan
│
├── data/
│   └── highscore.txt          # Penyimpanan skor tertinggi
│
└── requirements.txt           # Dependensi Python
```

Gambar 2. Struktur Direktori Proyek

Berikut adalah penjelasan singkat struktur direktori proyek **Flappy Bird Web** dengan format file dan fungsinya:

- **`main.py`**: Server FastAPI untuk menjalankan aplikasi, mengatur rute utama, menghubungkan WebSocket, dan menyajikan file statis seperti `` `index.html` ``.
- **`app/game_logic.py`**: Mengelola logika permainan utama, termasuk fisika burung, gerakan pipa, skor, dan status permainan (*preview*, *playing*, *paused*, *game over*).
- **`app/face_detection.py`**: Menangani deteksi wajah menggunakan MediaPipe untuk melacak posisi hidung dan memetakannya ke posisi burung dalam permainan.
- **`app/websocket_handler.py`**: Mengelola komunikasi WebSocket antara klien dan server, termasuk pemrosesan frame video, pembaruan status permainan, dan perintah seperti *start/pause/restart*.
- **`static/index.html`**: Antarmuka utama permainan, berisi elemen HTML seperti canvas untuk rendering permainan, video untuk deteksi wajah, dan tombol kontrol.
- **`static/game.js`**: Logika permainan sisi klien, mengatur rendering grafis, interaksi pengguna (tombol atau kamera), komunikasi WebSocket, dan pembaruan UI.

- **`static/assets`**: Folder yang menyimpan aset seperti sprite burung, pipa, latar belakang, dan musik untuk digunakan dalam permainan.
- **`data/highscore.txt`**: File untuk menyimpan dan membaca skor tertinggi pemain.
- **`requirements.txt`**: Daftar dependensi Python yang diperlukan untuk menjalankan aplikasi, seperti FastAPI, OpenCV, dan MediaPipe.

3.2.2. Implementasi Deteksi Hidung

Pada tahap ini, MediaPipe Face Mesh digunakan untuk mendeteksi titik-titik kunci pada wajah pengguna, dengan fokus pada titik hidung (*index landmark 1*) untuk mengontrol pergerakan vertikal burung dalam permainan. Posisi vertikal hidung (koordinat y) diambil dari hasil deteksi, dikalibrasi untuk memetakan rentang gerakan pengguna ke posisi burung (0.5 hingga 8.5 pada grid permainan), dan dismoothing menggunakan rata-rata bergerak (*moving average*) untuk mengurangi *jitter*. Data posisi hidung ini diproses di sisi *server* (face_detection.py) dan dikirim secara *real-time* melalui WebSocket (`websocket_handler.py`) ke klien (`game.js`) untuk memperbarui posisi burung di canvas permainan, memastikan sinkronisasi yang responsif dengan gerakan pemain.

3.2.3. Integrasi Permainan

Tahap ini mengintegrasikan logika permainan Flappy Bird pada sisi dengan sistem deteksi posisi hidung dari sisi. Komunikasi *real-time* diimplementasikan melalui WebSocket untuk mengirimkan posisi hidung yang telah dipetakan ke posisi burung (dengan rentang 0.5 hingga 8.5 pada grid permainan) dari *server* ke *client*, memastikan gerakan burung merespons perubahan posisi wajah pengguna secara instan. Di sisi backend (`game_logic.py`), mekanisme perhitungan skor diimplementasikan dengan menambah 10 poin setiap kali burung melewati celah pipa tanpa tabrakan, dan skor tertinggi disimpan ke file lokal `data/highscore.txt` melalui fungsi save_highscore dan read_highscore. FastAPI (`main.py`) mengelola komunikasi antara frontend dan backend, termasuk pembaruan status permainan seperti skor, jumlah pipa yang dilewati, dan kondisi game over, yang dikirimkan ke klien untuk diperbarui pada antarmuka.

3.2.4. Skenario Pengujian Model (Eksplorasi)

Pengujian dilakukan untuk mengevaluasi kinerja model pra-latih (pre-trained model) MediaPipe Face Mesh dalam berbagai kondisi pencahayaan dan sudut wajah. Tujuan dari skenario ini adalah

untuk memastikan stabilitas dan akurasi deteksi titik hidung yang menjadi input utama kendali permainan. Setiap skenario diuji menggunakan cuplikan video dengan kondisi tertentu untuk mensimulasikan penggunaan nyata.

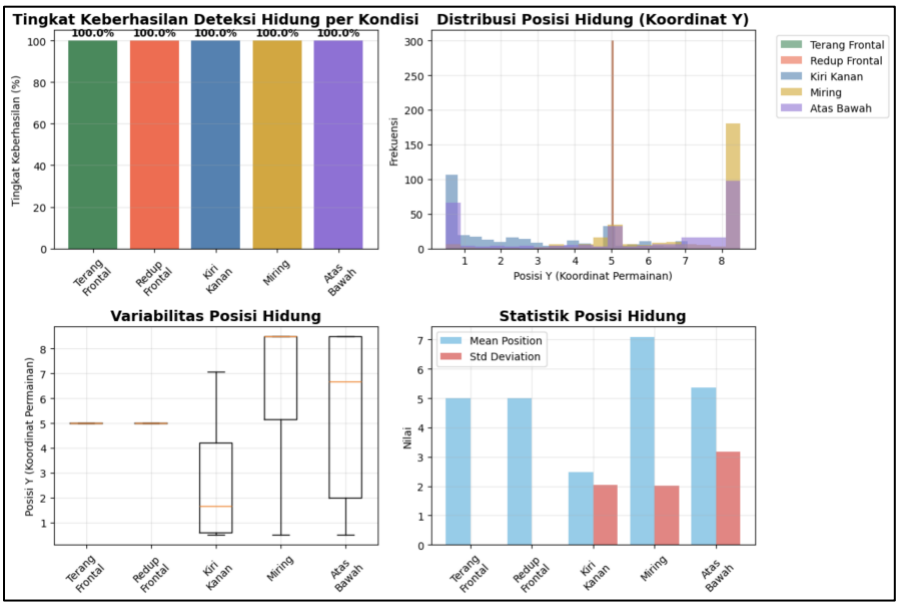
Adapun variabel uji yang dieksplorasi antara lain:

1. Terang-Frontal, pengujian dilakukan dengan pencahayaan terang yang datang langsung dari depan wajah pengguna. Kondisi ini merepresentasikan lingkungan ideal dan digunakan sebagai *baseline* performa deteksi.
2. Redup-Frontal, pengujian ini dilakukan dengan pencahayaan redup dari arah depan. Tujuannya untuk menguji sensitivitas model terhadap kekurangan cahaya dalam mendeteksi titik-titik wajah.
3. Kiri-Kanan pengujian ini mengkondisikan wajah pengguna yang diputar ke arah kiri dan kanan (*yaw rotation*) tanpa mengubah posisi kamera. Tujuannya adalah menguji kemampuan model mendeteksi titik hidung ketika wajah tidak menghadap langsung ke kamera.
4. Miring, pengguna memiringkan wajah ke kiri atau ke kanan (*head tilt*). Skenario ini menguji seberapa baik model mengenali posisi hidung saat orientasi wajah tidak sejajar secara vertikal.
5. Atas-Bawah, dalam skenario ini, pengguna menunduk atau mendongak. Hal ini bertujuan untuk mengamati kestabilan deteksi titik hidung ketika wajah tidak lagi sejajar horizontal terhadap kamera.
6. Performa Life-Time (*Real-Time Responsiveness*), pengujian dilakukan secara langsung (*live*) untuk melihat seberapa responsif dan konsisten model dalam membaca gerakan hidung secara *real-time* selama sesi permainan berlangsung. Metode ini juga mencakup uji beban sistem secara ringan pada perangkat pengguna (*latency*, *delay*, dan performa *frame-rate*).

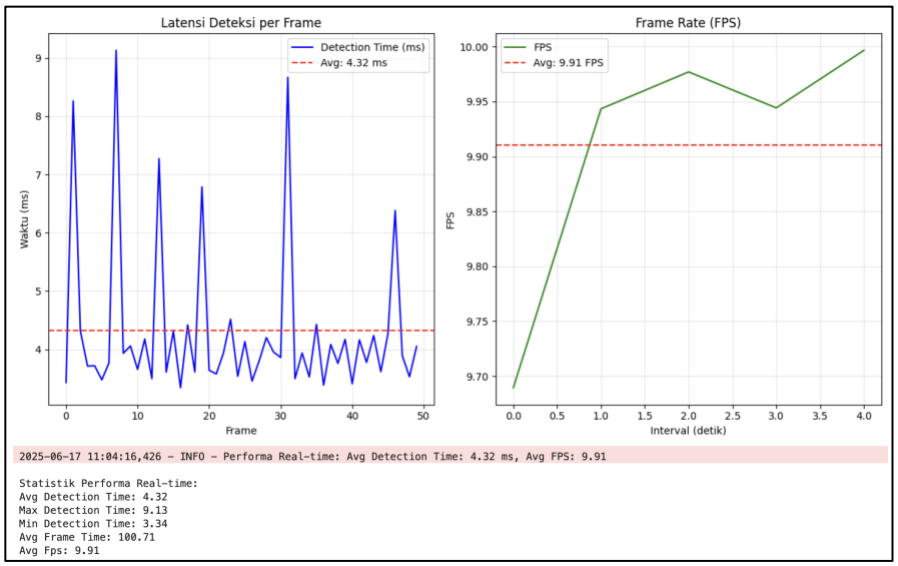
3.3. Analisis Hasil

3.3.1. Evaluasi Skenario Pengujian

Berdasarkan hasil pengujian skenario menggunakan model pra-latih MediaPipe Face Mesh, kinerja deteksi titik hidung dievaluasi dalam berbagai kondisi pencahayaan dan sudut wajah. Berikut adalah ringkasan hasil berdasarkan grafik:



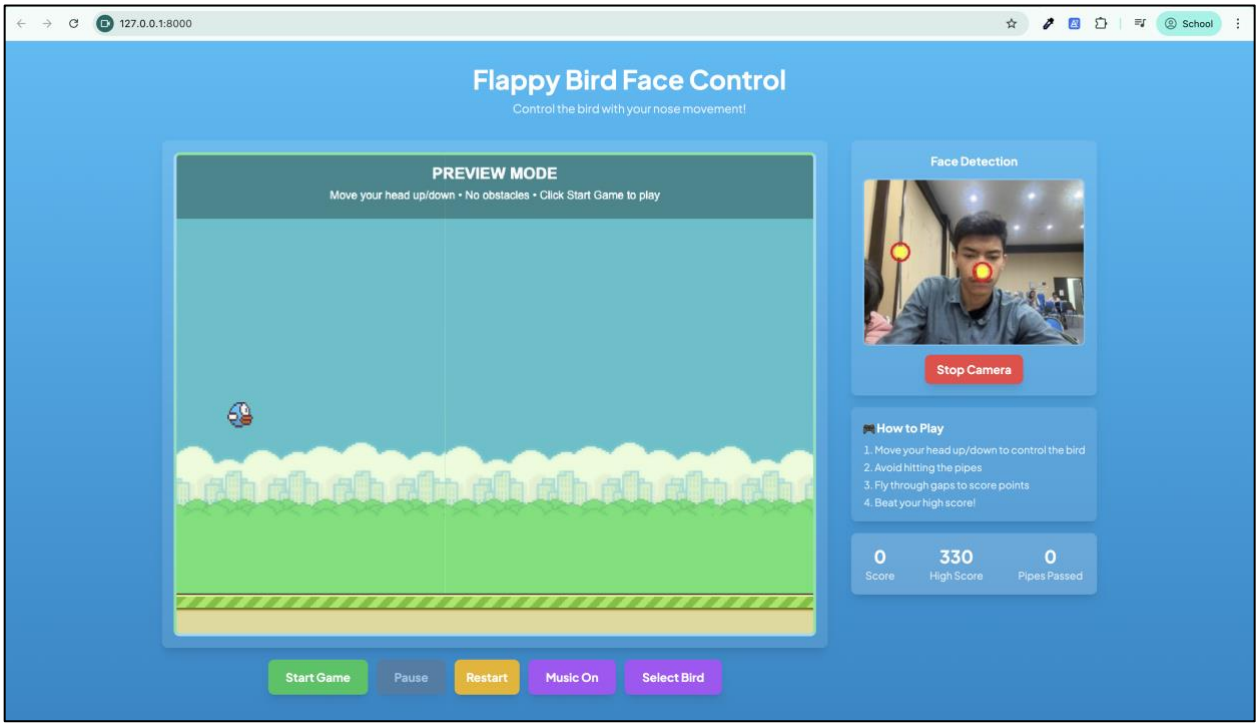
Gambar 3 Grafik Kinerja Model di Berbagai Kondisi



Gambar 4 Grafik Performa Real-time

Berdasarkan kedua grafik tersebut dapat disimpulkan bahwa, model *pre-train* MediaPipe Face Mesh menunjukkan performa luar biasa dengan tingkat keberhasilan 100% di semua kondisi yang diuji, didukung oleh distribusi posisi hidung yang stabil dan variabilitas rendah. Performa real-time dengan latensi rendah (di bawah 10 milisecond) dan FPS rata-rata sekitar 9.91 menjamin pengalaman permainan yang responsif, meskipun ada ruang untuk optimasi lebih lanjut pada kondisi "Atas-Bawah" yang menunjukkan deviasi standar lebih tinggi. Hasil ini mendukung penggunaan model ini dalam lingkungan nyata dengan rekomendasi untuk memastikan pencahayaan yang memadai dan orientasi wajah yang bervariasi.

3.3.2. Overall System



Sistem Flappy Bird Face Control menggabungkan deteksi wajah dengan permainan klasik Flappy Bird untuk menciptakan pengalaman bermain yang interaktif. Game ditampilkan di tengah layar dalam mode preview, di mana burung biru melayang dan dikendalikan oleh gerakan kepala pengguna berdasarkan posisi hidung yang dideteksi secara real-time menggunakan MediaPipe Face Mesh. Pengguna dapat memulai permainan dengan tombol "Start Game", serta mengakses kontrol tambahan seperti jeda, restart, musik, dan pilihan warna burung. Di sisi kanan, panel Face Detection menampilkan video dari kamera dengan penanda posisi hidung, serta tombol "Stop Camera" dan panduan singkat cara bermain. Skor, skor tertinggi, dan jumlah pipa yang dilewati diperbarui secara *real-time*.

Di belakang layar, sistem menggunakan FastAPI untuk logika permainan, deteksi wajah, dan komunikasi WebSocket. Modul `game_logic.py`, `face_detection.py`, dan `websocket_handler.py` mengatur proses utama, sementara *frontend* diatur oleh `game.js` dan `index.html`. Skor tertinggi disimpan secara lokal di `data/highscore.txt`.

BAB IV
PENUTUP

4.1. Kesimpulan

Berdasarkan implementasi dan analisis yang telah dilakukan, dapat disimpulkan bahwa proyek pengembangan game Flappy Bird berbasis kontrol wajah berhasil dikembangkan dengan memanfaatkan teknologi deteksi wajah menggunakan MediaPipe Face Mesh. Sistem ini mampu menerjemahkan gerakan vertikal hidung pengguna sebagai kontrol utama dalam permainan Flappy Bird secara *real-time* dan responsif.

Model *pre-trained* MediaPipe Face Mesh terbukti sangat akurat dan stabil dalam mendeteksi titik hidung di berbagai kondisi pencahayaan dan sudut wajah, mencapai tingkat keberhasilan 100% pada sebagian besar skenario pengujian dengan performa *real-time* yang stabil (rata-rata FPS ±9.91 dan latensi <10 ms).

Integrasi sistem yang menggunakan FastAPI sebagai backend untuk logika permainan dan WebSocket memungkinkan pengiriman data posisi hidung secara *real-time* dari kamera ke permainan, sehingga gerakan burung dapat dikontrol secara instan oleh gerakan hidung pengguna. Penyimpanan skor tertinggi secara lokal juga berhasil diterapkan, menjadikan game ini sepenuhnya *offline.* Secara keseluruhan, proyek ini berhasil menunjukkan penerapan konsep interaksi berbasis wajah dalam konteks hiburan, sekaligus melatih kemampuan dalam membangun sistem Flappy Bird berbasis kontrol wajah menggunakan MediaPipe Face Mesh.

4.2. Saran

Berdasarkan hasil analisis dan implementasi, beberapa saran untuk pengembangan selanjutnya adalah:

1. Penambahan Fitur Kalibrasi: Pengembangan fitur kalibrasi personal yang memungkinkan pengguna untuk menyesuaikan sensitivitas dan range deteksi sesuai dengan karakteristik wajah dan preferensi masing-masing pengguna akan meningkatkan user experience secara signifikan.
2. Eksplorasi Kontrol Tambahan: Pengembangan lebih lanjut dapat dilakukan untuk mengeksplorasi penggunaan fitur wajah lainnya sebagai kontrol tambahan, seperti kedipan mata untuk pause/resume, atau gerakan mulut untuk fitur khusus dalam permainan.

3. Implementasi Multiplayer dan Cloud Integration: Untuk pengembangan jangka panjang, sistem dapat diperluas dengan fitur multiplayer online dan integrasi cloud untuk leaderboard global, yang akan memerlukan pengembangan sistem komunikasi real-time yang lebih kompleks.
4. Pengujian pada Perangkat Mobile: Mengingat popularitas gaming mobile, disarankan untuk melakukan adaptasi dan pengujian sistem pada platform mobile (Android/iOS) untuk memperluas jangkauan pengguna dan mengevaluasi performa pada perangkat dengan keterbatasan sumber daya.

DAFTAR PUSTAKA

Fette, I. (2011). *The WebSocket Protocol*. RFC 6455. Retrieved June 16, 2025, from https://www.greenbytes.de/tech/webdav/rfc6455.pdf

Google AI. (n.d.). *MediaPipe solutions guide*. Google AI. Retrieved June 9, 2025, from https://ai.google.dev/edge/mediapipe/solutions/guide?hl=id

Google. (n.d.). *mediapipe 0.10.9*. PyPI. Retrieved June 9, 2025, from https://pypi.org/project/mediapipe/

Martinez, H. *Getting Started with Python and FastAPI: A Complete Beginner's Guide*. PyImageSearch Retrived June 16, 2025 from https://pyimg.co/fvjye

P2K STEKOM. (n.d.). *Flappy Bird. Ensiklopedia P2K*. Retrieved June 16, 2025, from https://p2k.stekom.ac.id/ensiklopedia/Flappy_Bird

M. H., Pinandito, A., & Fanani, L. (2018). *Perbandingan Websocket pada Komunikasi Aplikasi Perpesanan Berbasis Android Menggunakan Library AndroidAsync, Java Websocket, dan Nv Websocket Client*. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 2(11), 4999–5008.

Dixit, N., Shrivastava, V., Pandey, A., & Sharma, R. (2024). *Revolutionizing Web Design with Tailwind CSS: A Comprehensive Exploration*. International Journal of Research Publication and Reviews, 5(5), 4198–4204.

Yusuf Azhari, R. (2022). *Web Service Framework: flask dan fastAPI*. *Technology and Informatics Insight Journal, 1*(1). Retrieved June 17, 2025, from https://doi.org/10.32639/tiij.v1i1.54