



Program Studi Teknik Informatika Institut Teknologi Sumatera

LAPORAN TUGAS BESAR SISTEM TEKNOLOGI MULTIMEDIA

Voice Free Throw Basketball Game

Nama Mahasiswa : Reynaldi Cristian sssssSimamora
NIM : 122140116
Program Studi : Teknik Informatika
Mata Kuliah : Sistem Teknologi Multimedia
Kode Mata Kuliah : IF25-40305
Dosen Pengampu : Martin C.T. Manullang, Ph.D.

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SUMATERA
2025**

Repository Github

<https://github.com/anzensirc/Voice-Free-Throw>

1 PENDAHULUAN

Perkembangan teknologi multimedia interaktif dalam beberapa tahun terakhir menunjukkan peningkatan signifikan, terutama dalam integrasi antara *computer vision*, pemrosesan audio, dan antarmuka pengguna berbasis kamera. Kemudahan akses terhadap perangkat seperti webcam dan mikrofon memungkinkan pengembangan aplikasi interaktif tanpa memerlukan perangkat keras khusus. Salah satu bentuk implementasi yang memanfaatkan teknologi tersebut adalah permainan berbasis gesture dan suara yang berjalan secara *real-time*.

Pada proyek ini, penulis mengembangkan sebuah permainan sederhana yang mensimulasikan aksi tembakan bola basket menggunakan kombinasi deteksi gesture tangan dan analisis intensitas suara. Sistem memanfaatkan *MediaPipe Hands* untuk mendeteksi posisi dan kondisi tangan pemain, seperti membuka atau menutup tangan, yang berfungsi sebagai pemicu aksi *shooting*. Selain itu, pemrosesan audio dilakukan untuk mengukur tingkat tekanan suara (*sound level*) sebagai parameter kekuatan lemparan, melalui analisis *Fast Fourier Transform* (FFT) dan perhitungan *Root Mean Square* (RMS).

Lintasan bola dalam permainan dihitung menggunakan model fisika sederhana, mencakup kecepatan awal, gravitasi, dan simulasi probabilitas skor berdasarkan tingkat akurasi yang dihasilkan dari suara pemain. Rendering visual permainan dilakukan menggunakan OpenCV, termasuk tampilan latar, pemain, ring basket, lintasan bola, indikator akurasi, dan skor permainan. Seluruh proses dijalankan dalam satu loop *real-time* yang menggabungkan input video, pemrosesan audio, pemodelan fisika, dan visualisasi hasil.

Batasan Percobaan

Implementasi dan percobaan pada proyek ini dibatasi oleh hal-hal berikut:

1. Sistem hanya mendeteksi satu tangan ($max_num_hands = 1$) menggunakan MediaPipe.
2. Pemrosesan audio dilakukan menggunakan metode FFT dan RMS tanpa algoritma pembelajaran mesin.
3. Simulasi lintasan bola menggunakan fisika sederhana dan tidak memakai *physics engine* kompleks.
4. Rendering visual sepenuhnya berbasis OpenCV dalam format 2D dengan resolusi tetap.
5. Percobaan dilakukan menggunakan perangkat dengan kamera dan mikrofon internal standar.
6. Interaksi pemain terbatas pada gesture membuka-menutup tangan dan tingkat suara.
7. Aset visual seperti latar dan objek permainan bersifat statis.

2 Dasar Teori

2.1 MediaPipe Hands

MediaPipe Hands merupakan model deteksi dan pelacakan tangan berbasis *machine learning* yang mampu mengidentifikasi lokasi *landmark* tangan secara real-time. Model ini bekerja dengan dua tahap, yaitu pendeteksian tangan menggunakan *palm detector*, dan pelacakan 21 titik sendi menggunakan model regresi. Setiap *landmark* direpresentasikan dalam bentuk koordinat (x, y) pada citra, yang kemudian dapat digunakan untuk mendeteksi gestur tangan seperti membuka atau menutup telapak.

Dalam implementasi program, MediaPipe Hands digunakan untuk mendeteksi kondisi tangan terbuka (digunakan untuk persiapan menembak) dan tertutup (digunakan sebagai pemicu aksi menembak bola). Proses ini berjalan pada setiap frame dan mengembalikan hasil deteksi yang dipetakan ulang ke koordinat layar.

2.2 Pemrosesan Audio Menggunakan FFT

Fast Fourier Transform (FFT) merupakan algoritma yang digunakan untuk mengubah sinyal audio dari domain waktu menjadi domain frekuensi. Pada sistem ini, FFT digunakan untuk membaca intensitas suara secara real-time, yang kemudian dikonversi menjadi nilai amplitudo rata-rata.

Besarnya amplitudo suara digunakan sebagai parameter untuk menentukan kekuatan tembakan bola basket. Semakin besar nilai amplitudo, semakin jauh bola akan melayang.

Secara matematis, FFT mendekomposisi sinyal $x[n]$ menjadi representasi frekuensi:

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N}$$

2.3 Gerak Parabola pada Simulasi Bola

Lintasan bola dalam permainan mengikuti persamaan gerak parabola sederhana. Perpindahan pada sumbu horizontal dan vertikal dihitung berdasarkan persamaan:

$$x(t) = x_0 + v_x t$$

$$y(t) = y_0 + v_y t + \frac{1}{2}gt^2$$

dengan:

- v_x adalah komponen kecepatan horizontal,
- v_y adalah komponen kecepatan vertikal,
- g adalah percepatan gravitasi buatan (nilai disederhanakan dalam program),
- t adalah waktu dalam satuan langkah per frame.

Model fisika yang digunakan bersifat sederhana tanpa *air resistance* atau rotasi bola, sesuai kebutuhan implementasi untuk real-time rendering yang ringan.

2.4 OpenCV untuk Rendering Visual

OpenCV (*Open Source Computer Vision Library*) adalah pustaka komputasi visual yang mendukung operasi manipulasi citra, menggambar objek 2D, dan pengolahan video secara real-time.

Dalam program, OpenCV digunakan untuk:

- menampilkan frame video kamera,
- menggambar objek seperti bola, ring, dan lintasan,
- mengatur posisi elemen permainan berdasarkan hasil deteksi MediaPipe.

Seluruh proses rendering dilakukan pada resolusi tetap agar performa tetap stabil, tanpa melakukan komposisi 3D maupun efek grafis kompleks.

3 IMPLEMENTASI

3.1 Struktur Folder Program

Struktur program permainan *Voice Free Throw* disusun menggunakan pendekatan modular untuk menjaga keterbacaan serta memudahkan pemeliharaan kode. Setiap komponen dipisahkan ke dalam berkas tersendiri sehingga alur kerja lebih jelas dan pengembangan dapat dilakukan secara terarah. Struktur direktori program digambarkan sebagai berikut:

```
Voice-Free-Throw/  
  assets/  
    (berisi sound effects dan background music)  
  
  screenshots/  
    (berisi dokumentasi hasil uji coba dan gameplay)  
  
  audio_player.py  
  audio_processor.py  
  ball.py  
  config.py  
  game_state.py  
  hand_tracker.py  
  kernel_video.py  
  renderer.py  
  reset_game.py  
  vft.py  
  requirements.txt  
  README.md  
  .gitignore
```

Adapun fungsi utama tiap berkas adalah sebagai berikut:

- **vft.py**: titik masuk utama yang menjalankan seluruh proses permainan.
- **hand_tracker.py**: menangani deteksi dan pelacakan tangan menggunakan MediaPipe Hands.
- **audio_processor.py**: memproses sinyal audio menggunakan FFT untuk memperoleh amplitudo suara.
- **ball.py**: mengatur fisika bola, termasuk posisi, kecepatan awal, dan lintasan parabola.
- **game_state.py**: mengatur status permainan, seperti kondisi siap menembak dan pencatatan skor.
- **renderer.py**: menangani proses rendering grafis pada OpenCV.
- **kernel_video.py**: mengambil frame kamera dan menghubungkannya ke pipeline deteksi.
- **audio_player.py**: memutar efek suara tembakan dan skor.
- **config.py**: menyimpan parameter global seperti ukuran jendela dan konstanta fisika.
- **reset_game.py**: melakukan reset kondisi dan posisi bola setelah tembakan selesai.

3.2 Instalasi dan Menjalankan Program

Berikut langkah-langkah untuk memasang dan menjalankan permainan:

1. Clone Repository

```
git clone https://github.com/anzensirc/Voice-Free-Throw.git
cd Voice-Free-Throw
```

2. Install Dependencies

Pastikan Python versi 3.12 atau versi antara 3.10 sampai 3.12 telah terpasang, karena MediaPipe belum mendukung Python 3.13 atau lebih baru. Jika menggunakan **conda**, buat dan aktifkan environment baru dengan versi Python yang kompatibel:

```
conda create -n vft_env python=3.12.9 -y
conda activate vft_env
```

Alternatif versi Python yang bisa digunakan:

```
# conda create -n vft_env python=3.11.7 -y
# conda create -n vft_env python=3.10.11 -y
```

Kemudian instal seluruh dependensi:

```
pip install -r requirements.txt
```

3. Menjalankan Game

```
python vft.py
```

Catatan: Jangan gunakan Python 3.13 atau versi lebih baru karena belum didukung MediaPipe, gunakan environment terpisah agar instalasi Python dan library tidak bentrok dengan sistem utama.

3.3 Alur Kerja Program

Alur kerja permainan terbagi dalam dua tahap utama: inisialisasi dan pemrosesan setiap frame. Pada tahap awal, sistem menyiapkan kamera, modul MediaPipe Hands, modul audio, state permainan, serta komponen fisika bola.

Setiap frame, proses berjalan sebagai berikut:

1. Kamera menangkap citra dan modul **hand_tracker** mendeteksi gesture open-close sebagai pemicu tembakan.
2. Mikrofon merekam suara dan modul **audio_processor** menghitung level amplitudo melalui FFT bandpass.
3. Jika gesture tembakan terdeteksi, bola dilepas dengan kekuatan berdasarkan level audio.
4. Modul **ball.py** melakukan update posisi bola menggunakan persamaan gerak parabola.
5. Frame digambar kembali oleh **renderer.py**: bola, ring, lintasan, serta tampilan skor dan waktu.
6. Sistem mengevaluasi tabrakan rim untuk menentukan *score* atau *miss*.
7. Setelah tembakan selesai, modul **reset_game.py** memperbarui status permainan dan menyiapkan tembakan berikutnya.

3.4 Implementasi Tiap Modul

3.4.1 Pelacakan Landmark Tangan

Modul `hand_tracker.py` menggunakan Mediapipe Hands untuk mendeteksi gesture. Proses utama mencakup:

- ekstraksi dan penggambaran *landmark* tangan,
- perhitungan bounding box,
- deteksi tangan terbuka dan tangan tertutup,
- identifikasi transisi *open* \rightarrow *close* \rightarrow *open* untuk memicu aksi tembakan.

Modul ini memungkinkan interaksi berbasis gesture secara real-time.

3.4.2 Pemrosesan Suara

Modul `audio_processor.py` membaca sinyal mikrofon secara real-time dan mengubahnya menjadi nilai level suara (0–100) sebagai dasar kekuatan tembakan. Proses yang dilakukan meliputi:

- **Akuisisi audio:** pengambilan potongan sinyal menggunakan PyAudio.
- **Filtering FFT:** menerapkan bandpass pada 300–3000 Hz untuk mengurangi noise.
- **Perhitungan energi:** energi suara dihitung menggunakan RMS.
- **Normalisasi:** nilai RMS dikonversi ke skala 0–100.
- **Smoothing:** hasil dirata-rata menggunakan buffer sehingga level lebih stabil.

Nilai akhir level suara digunakan dalam hubungan:

$$F = \alpha \cdot A,$$

dengan A adalah amplitudo suara terproses.

3.4.3 Simulasi Bola

Modul `ball.py` mensimulasikan lintasan bola menggunakan model gerak parabola. Proses utama yang diterapkan:

- **Inisialisasi posisi dan kecepatan:** dihitung dari selisih posisi awal dan target.
- **Faktor akurasi suara:** menentukan peluang masuk (*score*) dan penyimpangan lintasan.
- **Perhitungan gerak:** posisi diupdate setiap frame dengan

$$x(t) = x_0 + v_x t, \quad y(t) = y_0 + v_y t + \frac{1}{2} g t^2.$$

- **Interaksi dengan ring:** deteksi tabrakan rim dan keputusan *score* atau *miss*.
- **Penggambaran:** bola, bayangan, dan jejak lintasan digambar menggunakan OpenCV.

Hanya fungsi inti (`update`, `_setup_trajectory`, `draw`) yang digunakan dalam alur permainan.

3.4.4 Rendering Visual

Modul `renderer.py` bertugas menggambar seluruh elemen game ke layar menggunakan OpenCV. Proses utama meliputi:

- **Menggambar latar:** langit, tanah, papan, ring, dan net.
- **Menggambar pemain:** tubuh, baju, serta animasi sederhana.
- **Menggambar bola:** posisi aktual dan bayangan untuk efek kedalaman.
- **Informasi HUD:** skor, waktu tersisa, dan status tembakan.
- **Integrasi kamera:** frame kamera diproses lalu ditampilkan sebagai bagian layar.

Komponen visual ini dirender ulang setiap frame dengan kecepatan 60 FPS.

3.4.5 Manajemen Audio Permainan

Modul `audio_player.py` bertanggung jawab untuk memutar musik latar (BGM) dan efek suara (SFX). Sistem menggunakan *pygame mixer* untuk:

- memuat berkas audio secara aman,
- mengatur volume efek suara,
- menjalankan fungsi pemutaran seperti `play_score()`, `play_miss()`, dan `play_best()`.

Modul ini memastikan seluruh elemen audio permainan dapat berjalan tanpa mengganggu proses utama.

3.4.6 Manajemen State Permainan

Modul `game_state.py` mendefinisikan struktur data permainan menggunakan *dataclass*. Variabel yang disimpan meliputi:

- skor, jumlah miss, dan best score,
- status tembakan dan objek bola,
- waktu permainan dan penghitung sisa waktu,
- akurasi tembakan dan hasil tembakan terakhir.

Struktur ini menjadi sumber kebenaran (*source of truth*) bagi seluruh logika permainan.

3.4.7 Reset Status Permainan

Modul `reset_game.py` mengatur ulang seluruh variabel permainan untuk memulai sesi baru. Proses ini meliputi:

- penyimpanan *best score*,
- pengaturan ulang skor, waktu, dan jumlah tembakan,
- pemilihan target akurasi secara acak,
- inisialisasi ulang kondisi *game active* dan *game over*.

Fungsi ini memastikan permainan kembali ke keadaan awal tanpa konflik status.

3.4.8 Peningkatan Kualitas Video

Modul `kernel_video.py` meningkatkan kualitas frame kamera untuk membantu deteksi gesture. Tahapan yang dilakukan:

- reduksi noise menggunakan Gaussian Blur,
- penajaman melalui *unsharp masking*,
- penyesuaian kontras dan kecerahan,
- koreksi warna sederhana menggunakan metode *gray world*.

Hasil peningkatan ini membuat pelacakan tangan lebih stabil dan akurat.

3.5 Program Utama (`vft.py`)

Modul `vft.py` berfungsi sebagai inti permainan yang mengatur alur eksekusi setiap komponen. Pada saat program dijalankan, sistem melakukan inisialisasi modul audio, pelacakan tangan, renderer, state permainan, serta kamera. Setelah proses inisialisasi, program memasuki loop utama yang berjalan setiap frame.

Secara garis besar, tugas program utama adalah:

1. Mengambil frame kamera dan meningkatkan kualitas citra menggunakan `enhance_frame`.
2. Memperbarui waktu permainan, termasuk durasi tersisa dan status *game over*.
3. Menjalankan modul `hand_tracker` untuk mendeteksi gesture tembakan.
4. Membaca level suara dari `audio_processor` dan menghitung akurasi tembakan.
5. Membuat objek bola dan menjalankan simulasi fisika jika tembakan dilakukan.
6. Memproses hasil tembakan (*score* atau *miss*) serta memutar efek suara melalui `audio_player`.
7. Menggambar seluruh elemen permainan: latar, ring, pemain, bola, jejak lintasan, HUD, dan preview tangan.
8. Menangani input keyboard seperti mulai permainan, restart, atau keluar.

Dengan demikian, `vft.py` berperan sebagai pengendali utama yang mengoordinasikan seluruh modul—mulai dari pemrosesan visual, audio, fisika, hingga antarmuka permainan.

4 HASIL DAN PEMBAHASAN

Bab ini menyajikan hasil implementasi dari permainan *Voice Free Throw*, meliputi tampilan antarmuka, respons sistem terhadap input gesture dan suara (visual gelombang dan kernel video), serta keluaran visual dari proses simulasi bola. Seluruh pengujian dilakukan menggunakan perangkat laptop dengan kamera dan mikrofon internal.

4.1 Tampilan Antarmuka dan Komponen Visual

Setelah program dijalankan, sistem menampilkan jendela permainan dengan beberapa elemen visual utama, seperti latar belakang, ring basket, deteksi posisi tangan menggunakan MediaPipe, bola basket, lintasan tembakan, indikator akurasi, serta skor permainan. Tampilan awal menunjukkan bola dalam posisi diam sebelum menerima input dari pengguna.

Berikut adalah beberapa tampilan utama dalam permainan:

1. Start Menu

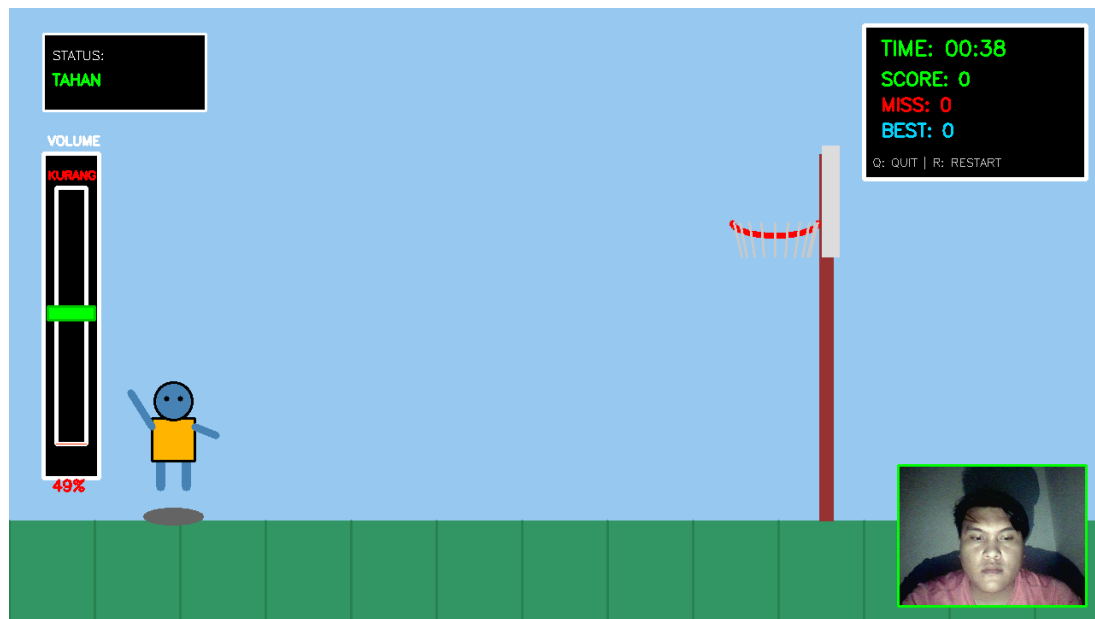
Menampilkan judul permainan, tombol mulai, serta instruksi singkat mengenai gesture dan kontrol suara.



Gambar 1: Tampilan Start Menu.

2. Holding Shot

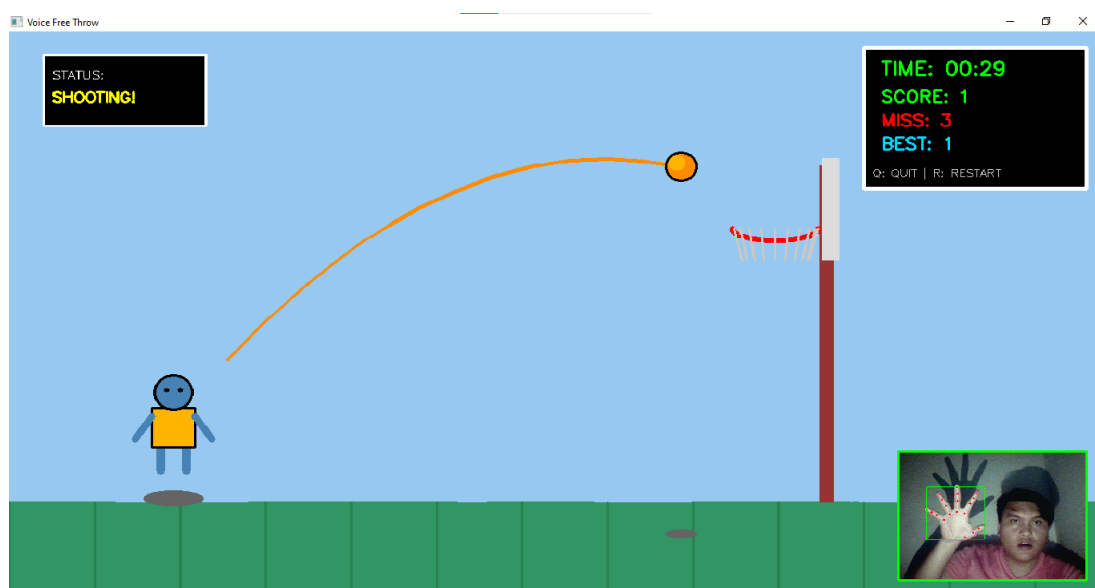
Bola berada dalam posisi awal, gesture menggenggam terdeteksi, dan indikator kekuatan suara mulai muncul.



Gambar 2: Tampilan saat menahan tembakan (Holding Shot).

3. Shooting

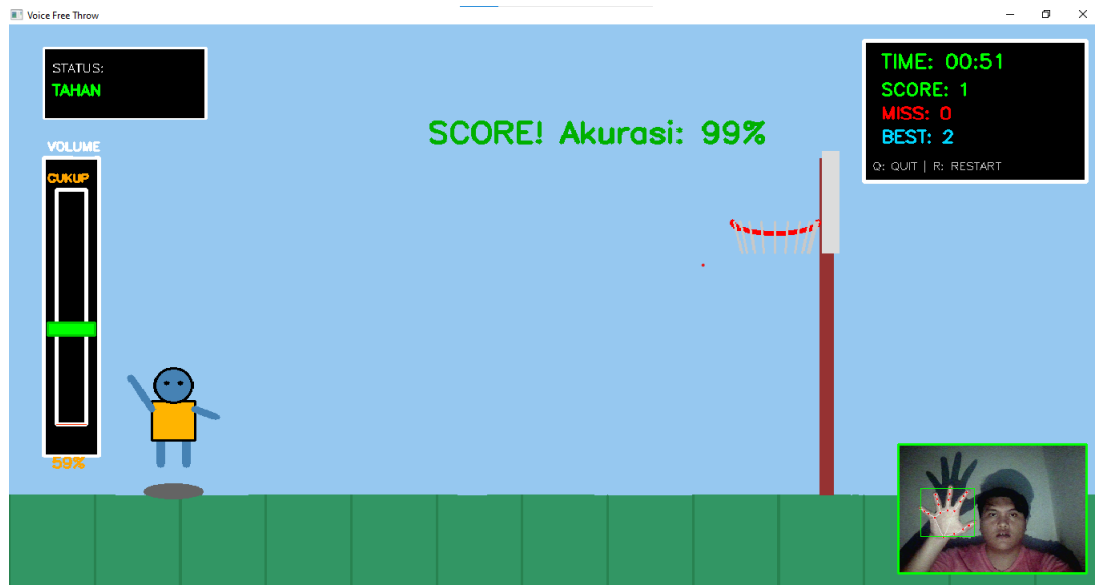
Bola dilepaskan dari genggamannya dan bergerak mengikuti lintasan parabola berdasarkan amplitudo suara.



Gambar 3: Tampilan proses menembak (Shooting).

4. Shooting Accuracy

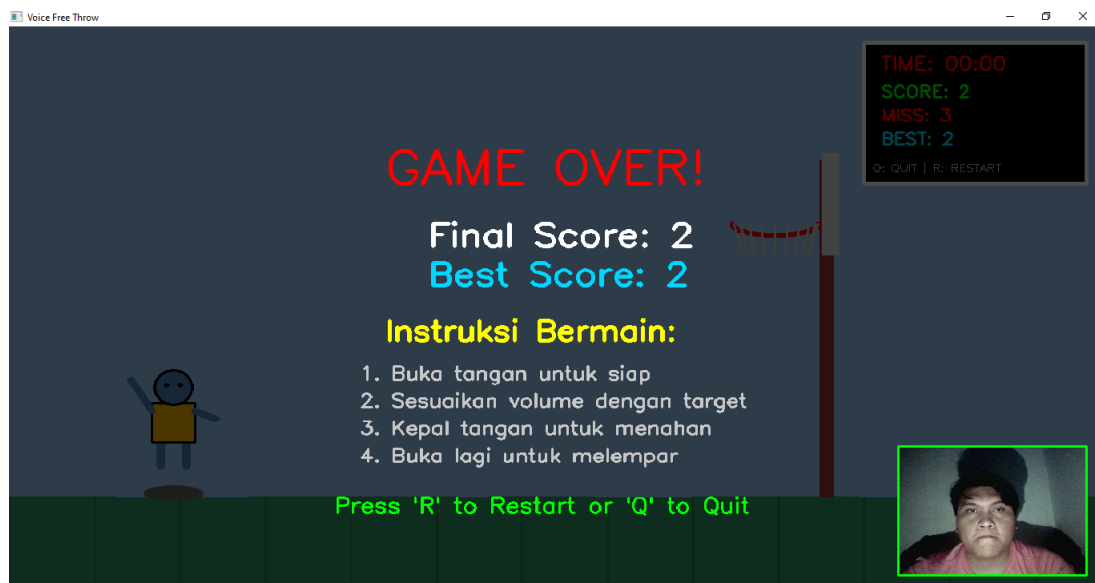
Sistem menilai apakah bola masuk ring dan menampilkan status tembakan (*Hit* atau *Miss*) beserta pembaruan skor.



Gambar 4: Tampilan evaluasi akurasi tembakan.

5. Game Over Menu

Ditampilkan saat percobaan habis atau sesi permainan berakhir, menampilkan skor akhir serta pilihan untuk mengulang permainan.



Gambar 5: Tampilan Game Over Menu.

4.2 Peningkatan Kualitas Video untuk Optimasi Deteksi Tangan

Untuk meningkatkan akurasi pendeteksian landmark tangan, sistem menerapkan tahap *video preprocessing* sebelum frame dikirim ke modul MediaPipe. Proses ini dilakukan melalui fungsi `enhance_frame()`, yang bertujuan untuk memperjelas kontur tangan dan menstabilkan kualitas citra pada kondisi cahaya yang bervariasi. Fungsi ini terdiri dari empat langkah inti:

1. **Gaussian Blur** Digunakan untuk mengurangi *noise* berfrekuensi tinggi sehingga MediaPipe lebih mudah mendeteksi bentuk tangan tanpa gangguan dari tepi berisik.

2. **Unsharp Mask (Sharpening)** Menerapkan kernel peningkat ketajaman gambar:

$$K = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Kernel ini memperjelas tepi tangan agar landmark lebih mudah dikenali.

3. **Penyesuaian Kontras dan Kecerahan** Peningkatan kontras adaptif dilakukan dengan:

$$f'(x) = \alpha f(x) + \beta$$

sehingga warna kulit lebih terdefinisi dan perbedaan objek-latar tampak lebih jelas.

4. **Auto-White-Balance (Gray World Approximation)** Menormalkan distribusi warna RGB agar frame tetap konsisten meskipun intensitas cahaya berubah.

Dengan empat proses ini, citra menjadi lebih bersih, tajam, dan seragam, sehingga meningkatkan stabilitas MediaPipe dalam mendeteksi 21 titik landmark tangan.

4.2.1 Integrasi dengan Deteksi Gesture

Frame yang telah ditingkatkan kualitasnya kemudian diproses oleh modul pendeteksi gesture. Beberapa fungsi inti yang digunakan adalah:

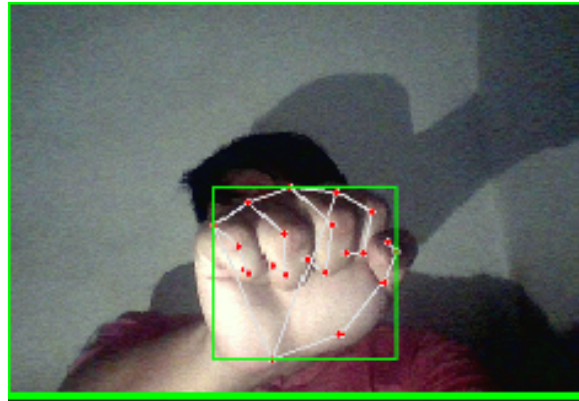
- **process(frame)** Mengevaluasi gesture berdasarkan landmark dan mendeteksi transisi *open* → *close* → *open* yang digunakan sebagai pemicu tembakan.
- **_is_hand_open()** Mengidentifikasi tangan terbuka dengan menganalisis posisi jari tip terhadap sendi PIP, serta posisi ibu jari.
- **_is_hand_closed()** Menentukan kondisi menggenggam berdasarkan kedekatan jari dengan pusat telapak.

Penggabungan antara kernel peningkatan video dan logika deteksi gesture membuat sistem lebih tahan terhadap perubahan cahaya, lebih stabil dalam menandai landmark, dan lebih konsisten dalam mendeteksi sinyal tembakan.

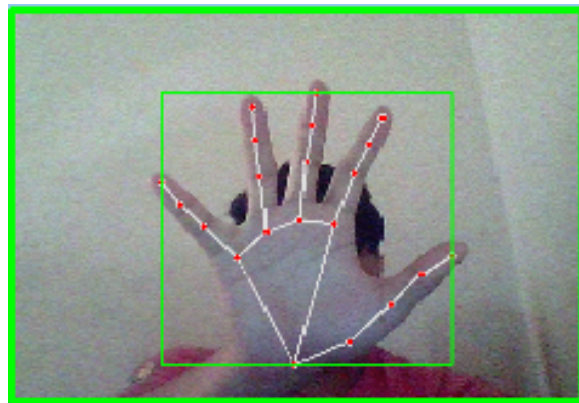
4.3 Hasil Deteksi Gesture Tangan

Sistem mendeteksi 21 titik landmark tangan menggunakan MediaPipe. Deteksi berjalan secara *real-time* dengan stabil, selama tangan pemain berada dalam jangkauan kamera. Hasil pengamatan menunjukkan:

- Sistem dapat membedakan keadaan tangan terbuka dan mengepal.
- Posisi jari telunjuk menjadi penentu lokasi awal bola.
- Deteksi tetap stabil meskipun terjadi variasi intensitas cahaya ringan.



Gambar 6: Deteksi Landmark Tangan Mengepal



Gambar 7: Deteksi Landmark Tangan Direntangkan

Gesture tangan menjadi faktor utama dalam memicu tembakan. Ketika tangan berpindah dari kondisi terbuka ke menggenggam, sistem menganggap pemain siap melakukan lemparan.

4.4 Hasil Pemrosesan Suara

Pemrosesan audio pada permainan dilakukan melalui modul `audio_processor.py`, yang membaca sinyal mikrofon secara real-time dan menghitung nilai amplitudo menggunakan *Fast Fourier Transform* (FFT). Nilai amplitudo ini kemudian digunakan sebagai parameter kekuatan tembakan bola.

4.4.1 Proses Akuisisi dan Pra-pemrosesan Audio

Sinyal audio direkam dengan laju cuplikan (*sample rate*) sebesar 44,100 Hz, yang merupakan standar industri untuk audio digital sehingga mampu menangkap rentang frekuensi suara manusia secara penuh. Setiap blok sinyal diproses dalam bentuk *frames* berukuran 2048 sampel, sebuah kompromi antara ketelitian analisis frekuensi dan respons waktu agar permainan tetap berjalan secara *real-time*.

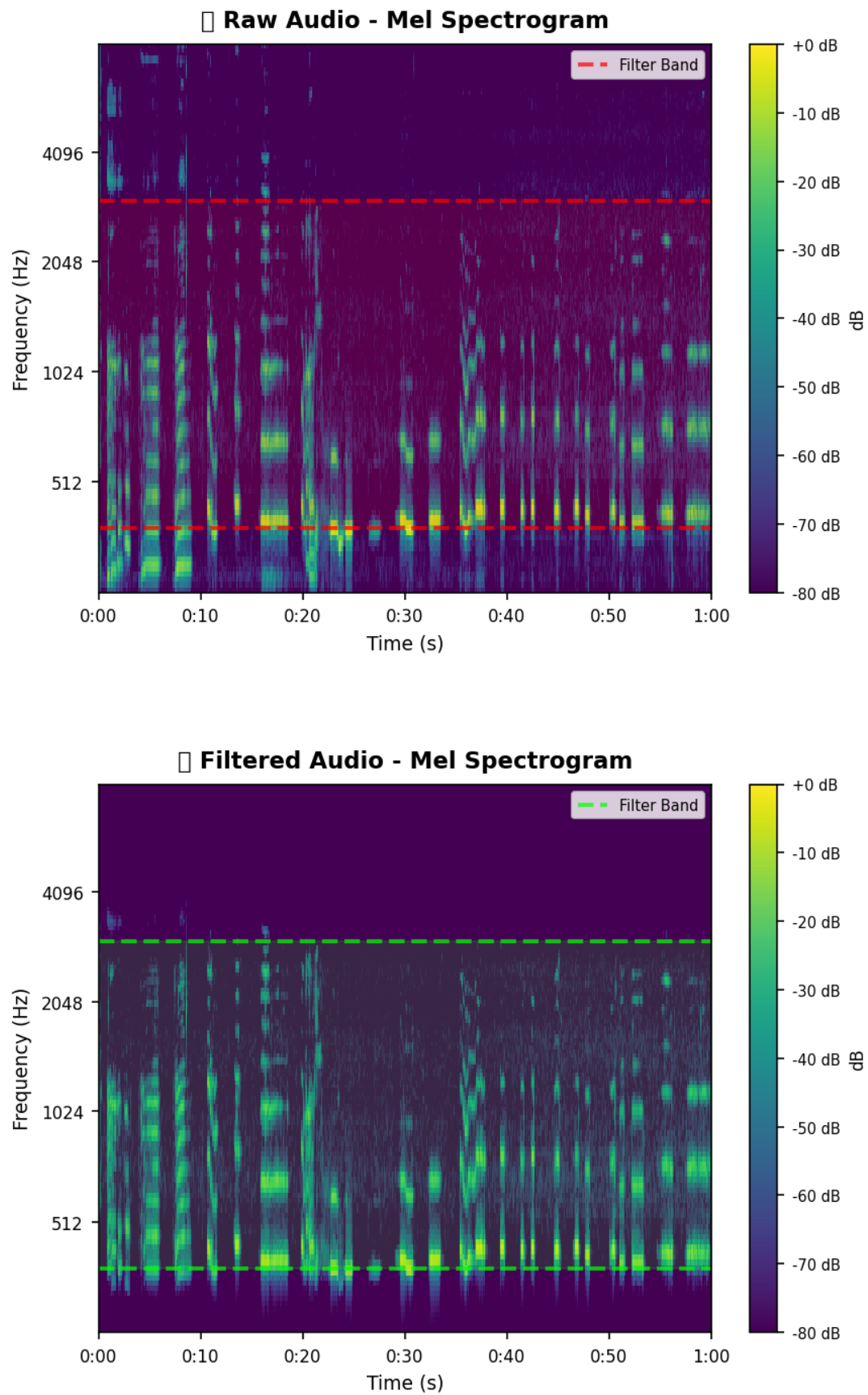
Sebelum dilakukan analisis frekuensi, sistem menerapkan filter pita (*band-pass filter*) pada rentang 300–3000 Hz. Rentang ini dipilih karena mencakup wilayah energi utama suara manusia, sehingga dapat menekan kebisingan frekuensi rendah (misalnya hum listrik 50/60 Hz) maupun gangguan frekuensi tinggi yang tidak relevan. Proses filtrasi ini menghasilkan sinyal yang lebih bersih dan terfokus, sebagaimana terlihat pada perbandingan spektrogram sebelum dan sesudah pemrosesan.

Selain FFT untuk estimasi amplitudo frekuensi, sistem juga menghitung nilai *Root Mean Square* (RMS) untuk memperoleh besaran energi sinyal dalam domain waktu. Nilai RMS digunakan untuk menstabilkan pembacaan amplitudo karena lebih tahan terhadap fluktuasi tajam dan *spikes* sesaat.

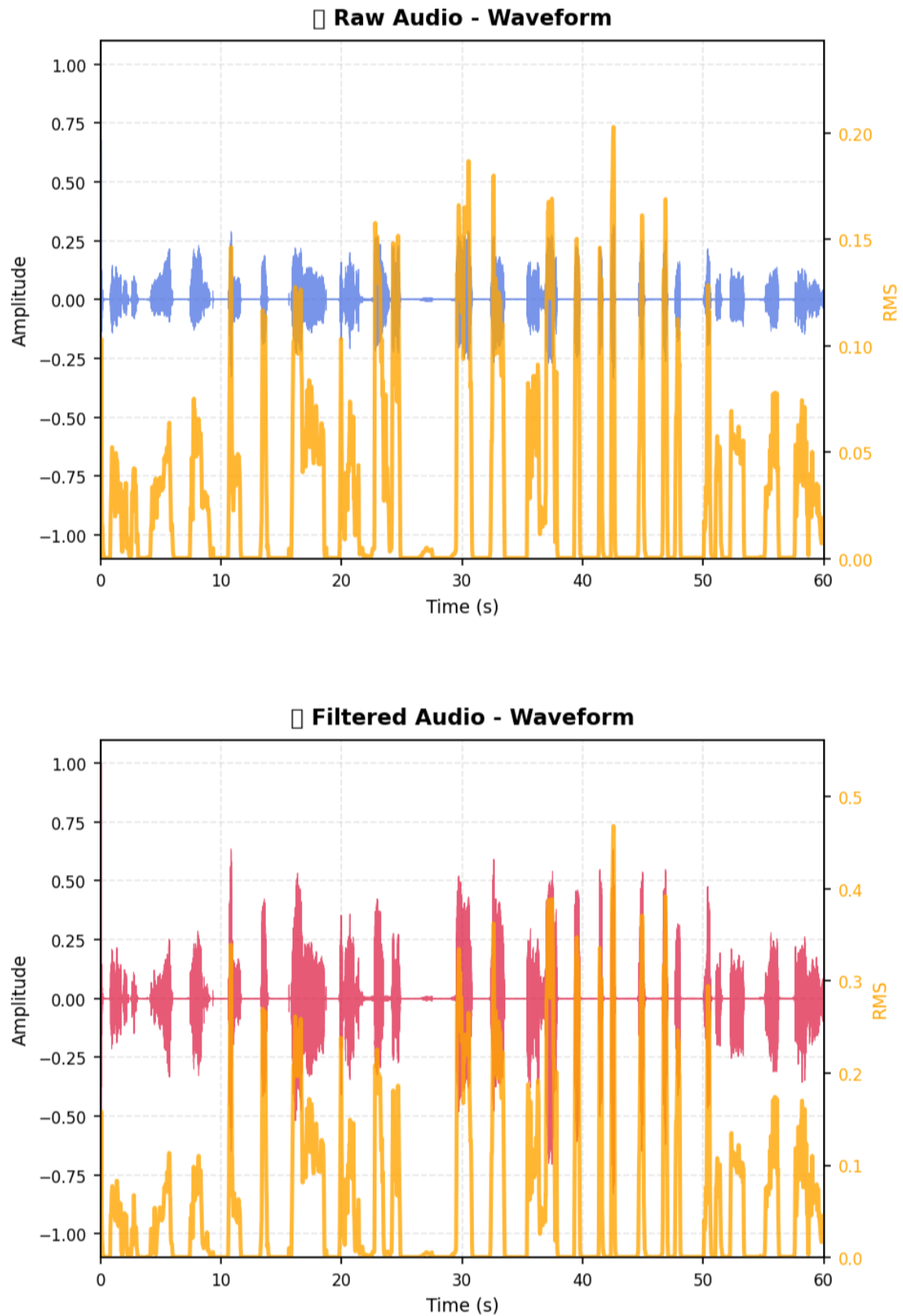
Penggunaan RMS pada kode `audio_processor.py` bertujuan memastikan bahwa kekuatan tembakan tidak dipengaruhi oleh noise impulsif, tetapi merepresentasikan intensitas suara pemain secara konsisten.

4.4.2 Analisis Spektrogram dan Waveform

Untuk pengambilan sampel audio ini menggunakan modul di luar kode untuk mengambil audio real time saat bermain. Sampel audio diambil dari 1 ronde permainan selama 1 menit, dan diberikan output plotnya. Modul ini menggunakan **PyAudio** untuk membaca sinyal mikrofon dan **NumPy** untuk melakukan FFT. Membandingkan waveform dan spectrogram sebelum dan sesudah proses filtrasi. Gambar 8 menunjukkan perbandingan *mel-spectrogram* sebelum dan sesudah proses filtrasi. Spektrogram awal memperlihatkan distribusi energi yang tersebar hingga frekuensi tinggi, sedangkan hasil filtrasi menunjukkan konsentrasi energi hanya pada pita relevan yang berkaitan dengan suara pengguna. Sementara itu, Gambar 9 memperlihatkan bentuk gelombang (*waveform*) beserta perhitungan RMS. Setelah filtrasi, amplitudo menjadi lebih halus dan responsivitas terhadap suara meningkat. Hal ini menghasilkan perhitungan kekuatan tembakan yang lebih konsisten.



Gambar 8: Perbandingan Spektrogram Sinyal Audio Sebelum dan Setelah Filtrasi

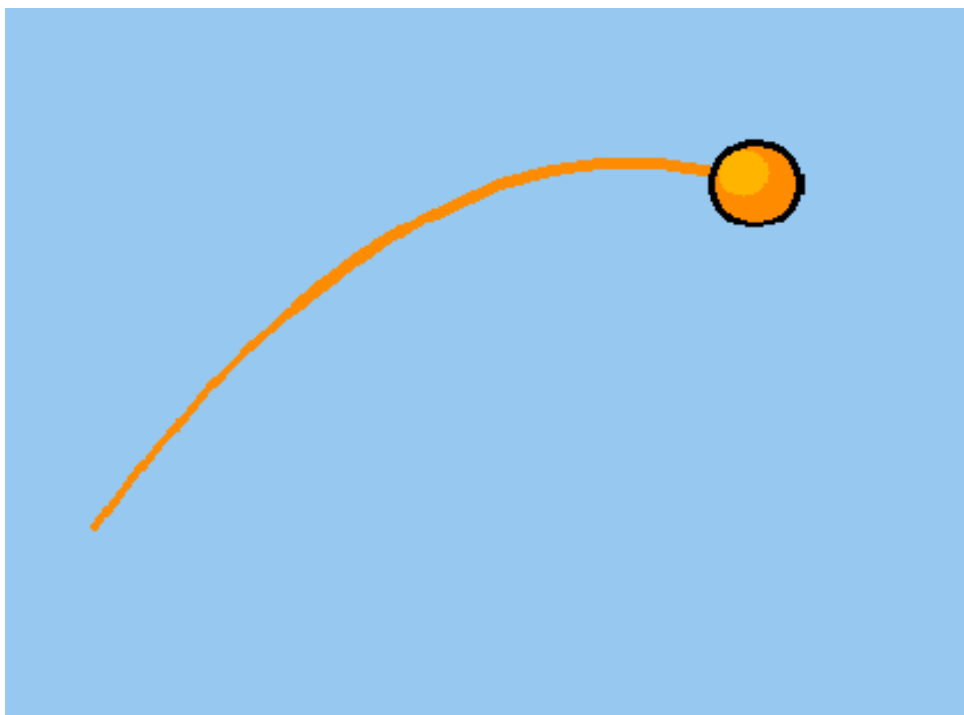


Gambar 9: Waveform Sinyal Audio Mentah

4.5 Hasil Simulasi Bola dan Akurasi

Simulasi lintasan bola dilakukan melalui model fisika sederhana sebagaimana diimplementasikan pada `ball.py`. Kecepatan awal bola ditentukan oleh selisih antara *target accuracy* dan nilai input suara pemain. Semakin kecil selisihnya, semakin besar nilai akurasi, dan semakin tinggi kecepatan awal bola. Posisi bola diperbarui setiap frame menggunakan persamaan gerak parabola dengan memper-timbangkan komponen kecepatan horizontal, vertikal, serta gravitasi. Untuk bar akurasi diambil dari audio, dan angka persentase melambangkan seberapa dekat akurasi dengan target.

Proses penilaian tembakan dilakukan dengan mendeteksi apakah posisi bola berada dalam jangkauan rim basket (*rim radius*). Ketika lintasan bola memasuki area tersebut, tembakan dinilai sebagai *score*; jika tidak mencapai area rim hingga bola jatuh di bawah ring, hasilnya adalah *miss*. Sistem juga memperbarui *target accuracy* secara acak setelah setiap tembakan untuk memberikan variasi tingkat kesulitan.



Gambar 10: Simulasi Lintasan Bola



Gambar 11: Bar Akurasi Suara

Beberapa poin penting dari hasil implementasi adalah sebagai berikut:

1. Sistem mampu menggabungkan deteksi tangan, pemrosesan suara, simulasi fisika, dan rendering visual secara *real-time*.
2. Respons permainan terhadap gesture dan suara bersifat dinamis dan konsisten.
3. Pengalaman bermain cukup stabil selama lingkungan pencahayaan dan kebisingan berada dalam kondisi wajar.
4. Seluruh modul bekerja sesuai ekspektasi: deteksi gesture, pengukuran suara, perhitungan lintasan, reset permainan, dan pencatatan skor.

5 KESIMPULAN

Proyek ini berhasil mengembangkan sistem permainan "Voice Free Throw" yang memanfaatkan kekuatan suara sebagai pengendali akurasi tembakan bola. Mekanisme pemrosesan audio menunjukkan kinerja yang stabil melalui kombinasi bandpass FFT, perhitungan RMS, dan proses smoothing, sehingga mampu menghasilkan estimasi kekuatan tembakan yang cukup responsif. Deteksi gesture menggunakan MediaPipe Hands juga berfungsi dengan baik dalam memberikan sinyal pemicu tembakan secara konsisten.

Walaupun masih terdapat beberapa kekurangan, khususnya pada akurasi pemrosesan audio dan kebutuhan pola gerakan tangan yang cukup spesifik sehingga respons tembakan tidak selalu optimal, sistem secara keseluruhan tetap dapat berjalan dengan baik dalam kondisi lingkungan yang sesuai.

Integrasi antara pemrosesan audio, perhitungan gerak parabola bola, dan visualisasi gameplay mampu bekerja secara selaras, menghasilkan simulasi tembakan yang akurat serta memberikan pengalaman permainan yang interaktif dan real-time.

6 SARAN

Beberapa pengembangan yang disarankan untuk penelitian atau proyek lanjutan adalah:

- Menambahkan beberapa konfigurasi sensitivitas suara agar dapat disesuaikan dengan karakteristik mikrofon.
- Menggunakan model pembelajaran mesin dan model proses audio yang lebih baik agar lebih akurat.
- Mengintegrasikan efek fisika yang lebih realistis pada bola dan ring.
- Menambahkan antarmuka UI/UX yang lebih dinamis dan 3D.

Lampiran

- **PyAudio (PortAudio Wrapper):** Digunakan untuk akuisisi audio streaming secara real-time dari mikrofon. <http://people.csail.mit.edu/hubert/pyaudio/>
- **NumPy FFT Module:** Modul `numpy.fft` digunakan untuk menerapkan *Fast Fourier Transform* dalam proses band-pass filtering. [NumPy FFT Documentation](#)
- **MediaPipe Hands Model:** Digunakan untuk pendeteksian pose tangan dan klasifikasi gestur (genggam / terbuka). [MediaPipe Hand Landmark Model](#)
- **OpenCV (cv2):** Digunakan untuk pembacaan kamera, prapemrosesan citra, dan menampilkan frame permainan. [OpenCV Documentation](#)
- **Audio Feedback (SFX/BGM):** Modul `pygame.mixer` digunakan untuk memutar efek suara tembakan, skor, miss, dan background music. [Pygame Mixer Documentation](#)
- **Voice Free Throw Repository:** <https://github.com/anzensirc/Voice-Free-Throw>
- **Tautan Video Demonstrasi:**
<https://youtu.be/UhmLmcsMPeM>
[Drive Video Demo](#)
- **Aset Audio:**
 1. miss.wav <https://pixabay.com/sound-effects/wrong-47985/>
 2. best.wav <https://pixabay.com/sound-effects/level-up-08-402152/>
 3. bgm.wav <https://pixabay.com/music/upbeat-game-mode-on-356552/>
 4. score.wav <https://elevenlabs.io/sound-effects/basketball-swish>
- **Bantuan Ai :**
 1. ChatGPT untuk membantu penulisan kode, brainstorming, dan dokumentasi. <https://chatgpt.com>
 2. Claude untuk inisiasi kode, dan struktur program. <https://claude.ai/>
 3. GitHub Copilot untuk saran kode otomatis dalam IDE.

Daftar Pustaka

References

- [1] P. R. R. Samal *et al.*, “Hand Gesture Recognition Using Mediapipe Framework,” *International Research Journal of Engineering and Technology (IRJET)*, vol. 9, no. 6, 2022. Available: <https://www.irjet.net/archives/V9/i6/IRJET-V9I6142.pdf>.
- [2] M. Telmem, N. Laaidi, and H. Satori, “The Impact of MFCC, Spectrogram, and Mel-Spectrogram on Deep Learning Models for Amazigh Speech Recognition System,” *International Journal of Speech Technology*, vol. 28, pp. 299–312, 2025. doi: [10.1007/s10772-025-10183-3](https://doi.org/10.1007/s10772-025-10183-3).
- [3] “Penggunaan OpenCV dalam Sistem Pengolahan Citra,” *Aviation Electronics, Information Technology, Telecommunications, Electricals, and Controls (AVITEC)*, vol. 7, no. 2, pp. 223–229, 2025. doi: [10.28989/avitec.v7i2.3132](https://doi.org/10.28989/avitec.v7i2.3132).
- [4] N. Nurwulandari, “Penerapan Game Angry Bird untuk Materi Gerak Parabola pada Pembelajaran Fisika,” *Jurnal Pendidikan: Riset & Konseptual*, vol. 2, no. 4, 2018. Available: http://journal.unublitar.ac.id/pendidikan/index.php/Riset_Konseptual. doi: [10.28926/riset_konseptual.v2i4.81](https://doi.org/10.28926/riset_konseptual.v2i4.81).