E-ISSN : 2988-1986
https://ejournal.cahayailmubangsa.institute/index.php/kohesi

Kohesi: Jurnal Multidisiplin Saintek
Volume 10 No 6 Tahun 2025

# UJI CEPAT SETELAN TRANSCODING BERBASIS CUPLIKAN: DESAIN DAN IMPLEMENTASI FFMPEG PROXY SANDBOX

**Vergino F. Maindoka [1], Arpen Patanduk [2], Andrew Evert Daniel Tondang [3], Wahyuni Fithratul Zalmi [4], Sherwin R. U Sompie [5] Ade Yusupa [6] Brave A Sugiarso [7]**
[1,2,3,4,5,6,7]Teknik Informatika , Fakultas Teknik, Universitas Sam Ratulangi
verginomaindoka026@student.unsrat.ac.id [1]
arpenpatanduk026@student.unsrat.ac.id [2] andrewtondang026@student.unsrat.ac.id [3]
wahyuni.fithratul.zalmi@unsrat.ac.id [4] aldo@unsrat.ac.id [5] ade@unsrat.ac.id [6]
brave@unsrat.ac.id [7]

## ABSTRACT

*The evolution of video compression standards such as H.264/AVC, H.265/HEVC, and VP9 requires a systematic exploration of transcoding settings to achieve a balance between visual quality, file size, and encoding time. However, testing directly on full-length videos using FFmpeg via the command line is time-consuming and error-prone. This study aims to design and implement FFmpeg Proxy Sandbox, a prototype tool for fast evaluation of transcoding settings based on short proxy clips with a simple graphical user interface. The development follows a sequential (waterfall-like) process, including requirements analysis, pipeline design, implementation of a Python engine that invokes FFmpeg, integration of a Tkinter-based GUI, functional testing, and transcoding experiments. The system generates multiple proxy clips from a single source video, performs transcoding according to recipes for H.264 (libx264), H.265 (libx265), and VP9 (libvpx-vp9), and automatically records objective quality metrics PSNR and SSIM, file sizes, and encoding times into a CSV summary file. Experimental results show that the tool clearly exposes the trade-offs between quality, size, and time: H.265 and VP9 configurations generally achieve higher file size reduction than H.264 at still high objective quality, but with longer encoding times, while recipes such as `x264-medium` provide a more balanced compromise for general use. Therefore, FFmpeg Proxy Sandbox can serve both as a practical tool for exploring transcoding settings and as a hands-on learning aid in multimedia systems contexts.*

***Keywords***: *FFmpeg, video transcoding, proxy clip, PSNR, SSIM, H.264, H.265, VP9, Tkinter.*

## ABSTRAK

Perkembangan standar kompresi video seperti H.264/AVC, H.265/HEVC, dan VP9 menuntut proses eksplorasi setelan

**Article History**
Received:
Reviewed:
Published:

Plagiarism Checker No 235
Prefix DOI :
**Copyright : Author**
**Publish by : Kohesi**

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License

transcoding yang sistematis agar diperoleh keseimbangan antara kualitas visual, ukuran berkas, dan waktu encoding. Namun, pengujian langsung pada video penuh menggunakan FFmpeg melalui command-line cenderung memakan waktu dan rentan kesalahan. Penelitian ini bertujuan merancang dan mengimplementasikan *FFmpeg Proxy Sandbox*, sebuah prototipe alat uji cepat setelan transcoding berbasis cuplikan (proxy) dengan antarmuka grafis sederhana. Metode pengembangan yang digunakan mengacu pada model sekuensial (waterfall), meliputi analisis kebutuhan, perancangan pipeline, implementasi engine Python yang memanggil FFmpeg, integrasi GUI berbasis Tkinter, serta pengujian fungsional dan eksperimen transcoding. Sistem menghasilkan beberapa proxy dari satu video sumber, menjalankan transcoding untuk resep H.264 (libx264), H.265 (libx265), dan VP9 (libvpx-vp9), serta secara otomatis menghitung metrik kualitas objektif PSNR dan SSIM, ukuran berkas, dan waktu proses ke dalam berkas ringkasan CSV. Hasil pengujian menunjukkan bahwa alat ini mampu menampilkan trade-off kualitas–ukuran–waktu secara jelas: setelan H.265 dan VP9 umumnya memberikan pengurangan ukuran berkas yang lebih besar dibanding H.264 pada kualitas objektif yang tetap tinggi, tetapi dengan durasi encoding yang lebih panjang, sedangkan resep seperti `x264-medium` memberikan kompromi yang seimbang untuk penggunaan umum. Dengan demikian, FFmpeg Proxy Sandbox dapat dimanfaatkan sebagai alat bantu eksplorasi setelan transcoding dan media pembelajaran praktis pada konteks sistem multimedia.

***Kata kunci: FFmpeg, transcoding video, proxy clip, PSNR, SSIM, H.264, H.265, VP9, Tkinter.***

## 1. PENDAHULUAN

### 1.1 Latar Belakang

Perkembangan teknologi multimedia dan distribusi video digital dalam beberapa tahun terakhir mendorong munculnya berbagai format dan standar kompresi, seperti H.264/AVC dan H.265/HEVC. Codec generasi baru seperti H.265 menawarkan efisiensi kompresi yang lebih tinggi—ukuran berkas dapat berkurang untuk kualitas visual yang sebanding—namun umumnya membutuhkan waktu encoding yang lebih lama dibanding H.264. Dalam praktiknya, pengembang sistem multimedia perlu mencari kombinasi setelan transcoding (codec, CRF, preset, dan parameter lainnya) yang paling seimbang antara kualitas gambar, ukuran berkas, dan durasi proses.

Di sisi lain, alur kerja produksi video modern banyak memanfaatkan proxy workflow, yaitu penggunaan salinan video beresolusi atau bitrate lebih rendah untuk keperluan pratinjau dan penyuntingan. Pendekatan ini membuat proses editing menjadi lebih ringan tanpa mengorbankan kualitas akhir, karena render final tetap menggunakan media asli. Konsep pemakaian "cuplikan" atau proxy ini tidak hanya relevan untuk editing, tetapi juga potensial untuk mempercepat eksperimen setelan transcoding: pengujian dapat dilakukan pada potongan video pendek yang representatif, bukan selalu pada keseluruhan durasi video sumber.

Penilaian objektif kualitas video hasil kompresi umumnya dilakukan menggunakan metrik seperti Peak Signal-to-Noise Ratio (PSNR) dan Structural Similarity Index (SSIM), yang membandingkan video hasil transcoding dengan video referensi. Kedua metrik tersebut banyak digunakan dalam penelitian kompresi dan kualitas video sebagai indikator perbedaan kualitas secara kuantitatif. Dalam konteks evaluasi sistem multimedia, informasi kuantitatif seperti PSNR, SSIM, ukuran berkas, dan waktu proses menjadi dasar untuk menilai apakah suatu konfigurasi codec layak digunakan dalam skenario nyata, misalnya streaming, penyimpanan arsip, atau distribusi konten beresolusi tinggi.

FFmpeg merupakan salah satu kerangka kerja multimedia open-source yang luas digunakan untuk perekaman, konversi, dan streaming multimedia, sekaligus menyediakan beragam codec video dan filter perhitungan kualitas. Walaupun sangat fleksibel, FFmpeg beroperasi melalui antarmuka baris perintah (command-line) yang menuntut pengguna memahami sintaks dan parameter yang cukup kompleks. Bagi mahasiswa, peneliti pemula, atau praktisi yang hanya ingin menguji beberapa kombinasi setelan transcoding dengan cepat, menyusun perintah FFmpeg secara manual untuk banyak percobaan menjadi pekerjaan yang repetitif, memakan waktu, dan rentan kesalahan pengetikan.

Akibatnya, proses eksplorasi konfigurasi transcoding sering kali tidak sistematis: setiap percobaan dijalankan secara ad-hoc, catatan parameter dan hasil uji tidak terdokumentasi dengan baik, dan sulit untuk mengulang atau membandingkan percobaan secara konsisten. Kondisi ini menunjukkan masalah utama yang ingin diselesaikan: belum tersedianya alat bantu yang ringkas dan terstruktur untuk mengelola rangkaian percobaan transcoding berbasis FFmpeg, mulai dari pembuatan

cuplikan video, pengujian beberapa resep sekaligus, hingga perekaman hasil ukur dalam bentuk data.

Dari sudut pandang pengembangan sistem multimedia interaktif, kebutuhan akan alat uji seperti ini sangat relevan. Mahasiswa dan peneliti tidak hanya membutuhkan hasil kompresi, tetapi juga sebuah sistem yang memungkinkan mereka berinteraksi dengan parameter transcoding, mengamati perintah FFmpeg yang dijalankan, serta langsung melihat dampak konfigurasi tersebut dalam bentuk metrik dan data ringkasan. Sistem seperti ini dapat menjadi jembatan antara konsep teori kompresi video dengan praktik eksperimen nyata di laboratorium maupun tugas akhir.

### 1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah dalam penelitian ini adalah sebagai berikut:

1) Bagaimana merancang sebuah alat uji cepat berbasis FFmpeg yang memanfaatkan cuplikan video sebagai proxy untuk mengevaluasi berbagai setelan transcoding tanpa harus selalu mengolah keseluruhan video sumber?
2) Bagaimana mengintegrasikan perhitungan metrik objektif (PSNR dan SSIM), ukuran berkas, serta waktu proses ke dalam pipeline transcoding sehingga hasil uji terdokumentasi secara otomatis dalam bentuk data terstruktur yang siap dianalisis?
3) Bagaimana merancang dan mengimplementasikan antarmuka grafis yang mampu mempermudah pengguna dalam memilih video masukan dan parameter proxy, memilih dan menjalankan resep transcoding, mengamati perintah FFmpeg yang dijalankan, serta meninjau hasil pengujian dalam bentuk data terstruktur?

Rumusan masalah tersebut menegaskan bahwa fokus penelitian bukan hanya pada kompresi video itu sendiri, tetapi pada perancangan sistem uji yang terintegrasi, interaktif, dan dapat mendukung kegiatan eksperimen di bidang sistem multimedia.

### 1.3 Tujuan dan Kontribusi Penelitian

Tujuan dari penelitian dan pengembangan sistem ini adalah:

1) Mengembangkan prototipe alat uji cepat setelan transcoding berbasis cuplikan, yang dalam penelitian ini disebut FFmpeg Proxy Sandbox, yang mampu menghasilkan proxy klip dari video masukan, melakukan proses transcoding dengan beberapa resep yang didefinisikan dalam berkas konfigurasi, serta menghitung metrik objektif PSNR dan SSIM serta merekam ukuran berkas dan waktu encoding.
2) Mengimplementasikan antarmuka pengguna yang mengemas pipeline pembuatan proxy, transcoding, dan perhitungan kualitas ke dalam alur kerja yang lebih mudah dipahami dan dioperasikan, khususnya oleh pengguna yang belum terbiasa dengan FFmpeg secara langsung.
3) Menyajikan hasil uji beberapa resep codec (misalnya variasi H.264 dan H.265 dengan kombinasi CRF dan preset tertentu) dalam bentuk tabel dan visualisasi sederhana, serta menganalisis trade-off antara kualitas, ukuran berkas, dan durasi proses encoding berdasarkan data yang dihasilkan oleh sistem.

Secara khusus, kontribusi kebaruan (novelty) dari penelitian ini dalam konteks pengembangan sistem multimedia interaktif dapat dirangkum sebagai berikut:

1) Menggabungkan konsep proxy clip dan recipe-based transcoding ke dalam satu alur kerja otomatis, sehingga proses uji setelan transcoding dapat dilakukan dengan cepat dan berulang secara konsisten.
2) Menyediakan mekanisme pengukuran kualitas yang terintegrasi (PSNR, SSIM, ukuran berkas, dan waktu proses) serta ekspor hasil dalam bentuk ringkasan terstruktur (misalnya berkas CSV), yang langsung siap digunakan sebagai bahan analisis maupun pelaporan.
3) Merancang dan membangun antarmuka grafis ringan berbasis Tkinter yang tetap transparan terhadap perintah FFmpeg yang dijalankan, sehingga sistem dapat berfungsi sekaligus sebagai alat eksperimen dan media pembelajaran bagi pengguna dalam memahami hubungan antara parameter transcoding dan kualitas hasil video.

## 2. METODE PENELITIAN

### 2.1 Rancangan atau Desain Penelitian

Penelitian ini termasuk kategori rekayasa perangkat lunak dengan pendekatan eksperimen. Fokus utamanya adalah merancang dan mengimplementasikan alat bantu uji transcoding video, kemudian menggunakan alat tersebut untuk membandingkan beberapa setelan codec.

Proses pengembangan mengikuti model sekuensial (waterfall), yang membagi pekerjaan ke beberapa tahap: analisis kebutuhan, perancangan, implementasi, pengujian, dan evaluasi. Model ini dipilih karena alur kerja sistem cukup jelas dan linier, sehingga setiap tahap dapat didokumentasikan dengan baik dan mudah.

Hasil utama dari rancangan ini adalah:

1) Prototipe perangkat lunak FFmpeg Proxy Sandbox yang terdiri dari:
    a) engine berbasis Python yang berinteraksi dengan FFmpeg untuk membuat proxy clip, menjalankan transcoding berdasarkan resep, dan menghitung metrik kualitas;

    b) antarmuka grafis berbasis Tkinter yang menjadi media interaksi pengguna dengan pipeline tersebut.

2) Data hasil pengujian transcoding berupa berkas ringkasan yang berisi ukuran berkas, waktu encoding, serta nilai metrik PSNR dan SSIM untuk setiap kombinasi klip proxy dan resep transcoding.

### 2.2 Tahapan Pengembangan

#### 2.2.1 Analisis Kebutuhan

Tahap awal adalah mengidentifikasi kebutuhan fungsional dan non-fungsional sistem melalui:

- studi literatur kompresi gambar dan video, termasuk perbandingan codec H.264, H.265, dan VP9;
- kajian metrik kualitas video objektif, khususnya PSNR dan SSIM;
- telaah dokumentasi resmi FFmpeg terkait dukungan codec dan filter pengukuran kualitas;
- identifikasi kebutuhan praktikum atau penelitian di bidang sistem multimedia.

Dari sini dirumuskan kebutuhan utama:

- pembuatan beberapa proxy clip dari satu video sumber tanpa re-encoding;
- mekanisme resep transcoding yang fleksibel untuk mengelola kombinasi parameter codec;
- perhitungan otomatis PSNR dan SSIM, serta pencatatan ukuran berkas dan waktu proses;
- ekspor hasil pengujian ke berkas ringkasan CSV;
- antarmuka grafis yang menampilkan status proses serta perintah FFmpeg yang dijalankan.

### 2.2.2 Perancangan Sistem dan Pipeline

Pada tahap perancangan, sistem dibagi ke beberapa komponen utama:

1) Modul pembuatan proxy

Modul ini menerima video sumber, daftar waktu mulai, dan durasi cuplikan. Sistem menyusun perintah FFmpeg untuk mengekstrak potongan video (proxy clip) tanpa re-encoding (-c copy) sehingga proses

berlangsung cepat dan kualitas referensi tetap terjaga.

2) Modul resep transcoding (recipes.json)

Setelan transcoding disimpan dalam berkas JSON dengan struktur berikut:

```
{
"x264-low": { "codec": "libx264", "preset": "fast", "crf": 28 },
"x264-medium": { "codec": "libx264", "preset": "medium", "crf": 23 },
"x264-hq": { "codec": "libx264", "preset": "slow", "crf": 18 },

"x265-small": { "codec": "libx265", "preset": "medium", "crf": 30 },
"x265-balanced": { "codec": "libx265", "preset": "slow", "crf": 28 },
"x265-hq": { "codec": "libx265", "preset": "slower", "crf": 24 },

"vp9-balanced": { "codec": "libvpx-vp9", "crf": 32, "b:v": "0", "deadline": "good" },
"vp9-hq": { "codec": "libvpx-vp9", "crf": 28, "b:v": "0", "deadline": "best" }
}
```

a) Setiap ID resep menjadi kunci.
b) Nilainya adalah pasangan parameter FFmpeg yang relevan.
c) Pengguna dapat mengubah atau memodifikasi resep selama format JSON dan nama parameternya valid untuk FFmpeg.

3) Modul transcoding dan pengukuran kualitas

Untuk setiap kombinasi proxy dan resep, sistem:

a) menyusun dan menjalankan perintah FFmpeg sesuai parameter resep;
b) mengukur durasi proses;
c) menjalankan FFmpeg lagi untuk menghitung PSNR dan SSIM terhadap proxy asli.

4) Modul pencatatan hasil (summary)

Sistem mencatat identitas video atau proxy, ID resep, informasi codec,

ukuran berkas sebelum dan sesudah encoding, durasi proses, serta nilai PSNR dan SSIM ke dalam satu berkas CSV.

5) Modul antarmuka grafis
   Antarmuka Tkinter mengoperasikan tiga alur utama:

   a) Proxy & Test Recipes

   b) Apply Recipe to Single Full Video

   c) Apply Recipe to Multiple Full Videos

6) Antarmuka menyediakan konfigurasi input, daftar resep, pengaturan proxy, dan log perintah FFmpeg.

### 2.2.3 Implementasi Engine

Engine dikembangkan menggunakan Python. FFmpeg dipanggil sebagai proses eksternal, dengan argumen yang dibangun dari:

- jalur berkas input dan output,
- parameter waktu mulai dan durasi proxy,
- parameter pada resep transcoding.

Sebelum pipeline dijalankan, sistem memeriksa:

- ketersediaan FFmpeg,
- validitas recipes.json, termasuk tipe data parameter dan kelengkapan struktur.

### 2.2.4 Implementasi Antarmuka Grafis

Integrasi Tkinter dilakukan setelah engine baris perintah siap. Implementasi mencakup:

- jendela utama dengan area input, daftar resep, dan log;
- pengaitan tombol aksi ke fungsi engine;
- penggunaan threading agar proses tidak membuat antarmuka berhenti merespons.

Desainnya sederhana agar cocok sebagai alat eksperimen maupun media belajar. Seluruh kode sumber yang mencakup engine, antarmuka grafis Tkinter, struktur recipe, serta skrip pendukung tersedia dalam repositori GitHub: https://github.com/V3RC0/Proxy-Sanbox-FFMPEG

### 2.2.5 Pengujian Fungsional Sistem

Pengujian dilakukan untuk memastikan:

- recipes.json dapat dibaca dan kesalahan format ditangani,
- proxy clip dibuat sesuai parameter,
- transcoding H.264, H.265, dan VP9 berjalan sesuai resep,
- PSNR dan SSIM tercatat dalam berkas ringkasan,
- antarmuka grafis menampilkan status dan log secara benar.

## 2.3 Subjek dan Lokasi Penelitian

Objek penelitian terdiri dari:

1) Perangkat lunak FFmpeg Proxy Sandbox (engine Python dan antarmuka Tkinter).

2) Berkas video uji sebagai sumber proxy dan referensi kualitas.

3) Kumpulan resep transcoding dalam recipes.json yang mencakup codec H.264, H.265, dan VP9.

Penelitian tidak melibatkan responden manusia. Lokasi penelitian adalah lingkungan pengembangan penulis pada komputer pribadi dengan FFmpeg dan Python terpasang. Spesifikasi perangkat keras dan perangkat lunak dijelaskan pada bagian hasil dan pembahasan.

### 2.4 Teknik dan Instrumen Pengumpulan Data

Data diperoleh melalui keluaran sistem berdasarkan skenario transcoding.

1) Menentukan skenario uji
   Memilih video sumber dan membuat beberapa proxy clip, lalu menguji beberapa resep.

2) Menjalankan pipeline melalui antarmuka
   Sistem memanggil FFmpeg secara otomatis untuk membuat proxy, melakukan encoding, dan menghitung metrik.

3) Pencatatan ke berkas ringkasan
   Data seperti ukuran berkas, durasi proses, PSNR, dan SSIM disimpan dalam CSV.

4) Dokumentasi pendukung
   Log FFmpeg dan tangkapan layar antarmuka disimpan sebagai bukti pelaksanaan.

Instrumen utama: FFmpeg Proxy Sandbox, FFmpeg, dan aplikasi pengolah CSV.

### 2.5 Teknik Analisis Data

Data dianalisis menggunakan pendekatan deskriptif kuantitatif melalui langkah berikut:

1) Pengelompokan berdasarkan codec dan resep
   Data diurutkan berdasarkan keluarga codec dan ID resep.

2) Perbandingan kualitas dan efisiensi
   Analisis mencakup:

   a) PSNR dan SSIM sebagai indikator kualitas,

   b) perubahan ukuran berkas sebagai indikator kompresi,

   c) durasi encoding sebagai indikator beban komputasi.

3) Penyajian hasil
   Disajikan dalam bentuk tabel sederhana.

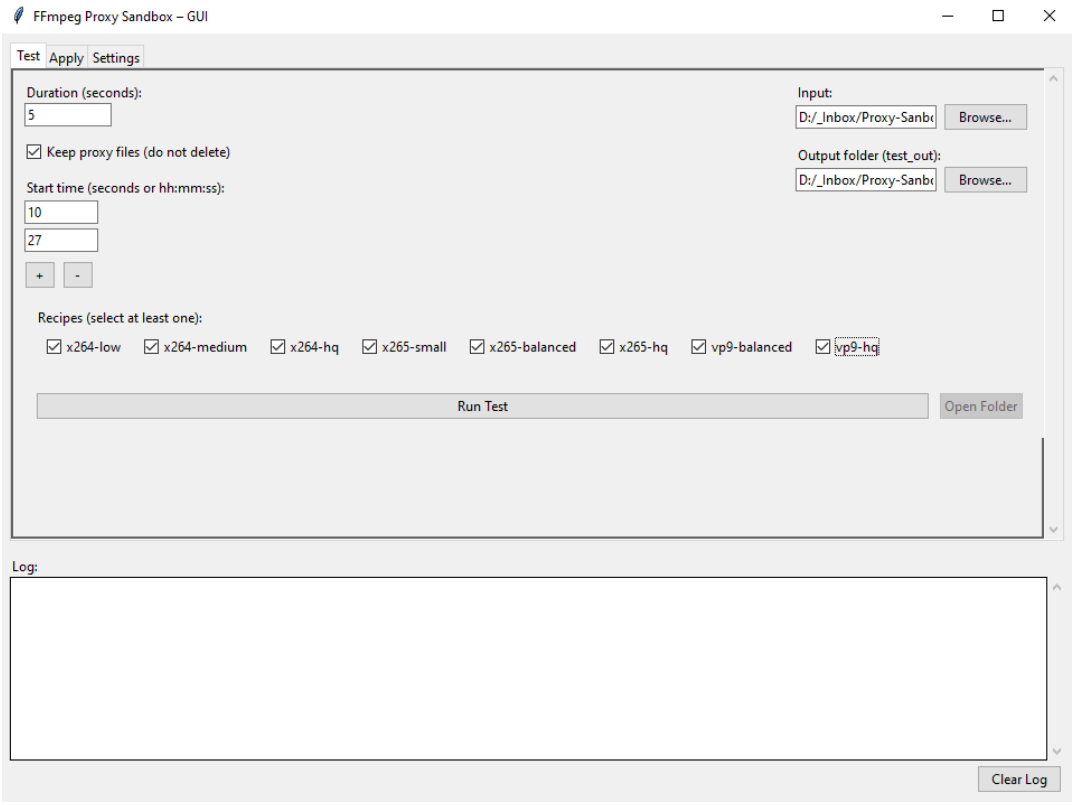
4) Interpretasi dan kaitan dengan literatur
   Pola hasil dibandingkan dengan temuan studi sebelumnya.

Analisis ini memberikan gambaran hubungan parameter transcoding terhadap kualitas dan efisiensi, sekaligus mengevaluasi bagaimana FFmpeg Proxy Sandbox mendukung eksplorasi tersebut.

## 3. HASIL DAN PEMBAHASAN

### 3.1 Implementasi Sistem dan Tampilan Antarmuka

Sistem yang dikembangkan diimplementasikan sebagai aplikasi desktop dengan tiga tab utama: Test, Apply, dan Settings. Pada tab Test, pengguna dapat memilih video masukan, menentukan durasi proxy, mengisi daftar waktu mulai (start time) cuplikan, memilih resep transcoding, serta menjalankan proses pengujian. Di bagian bawah terdapat area log yang menampilkan perintah FFmpeg dan pesan status, serta panel status yang menandai keberhasilan deteksi FFmpeg.




gambar 1.1 screenshot tampilan awal tab Test (sebelum input diisi dan sebelum proses dijalankan)

Setelah parameter diisi dan tombol Run Test di klik, aplikasi akan:

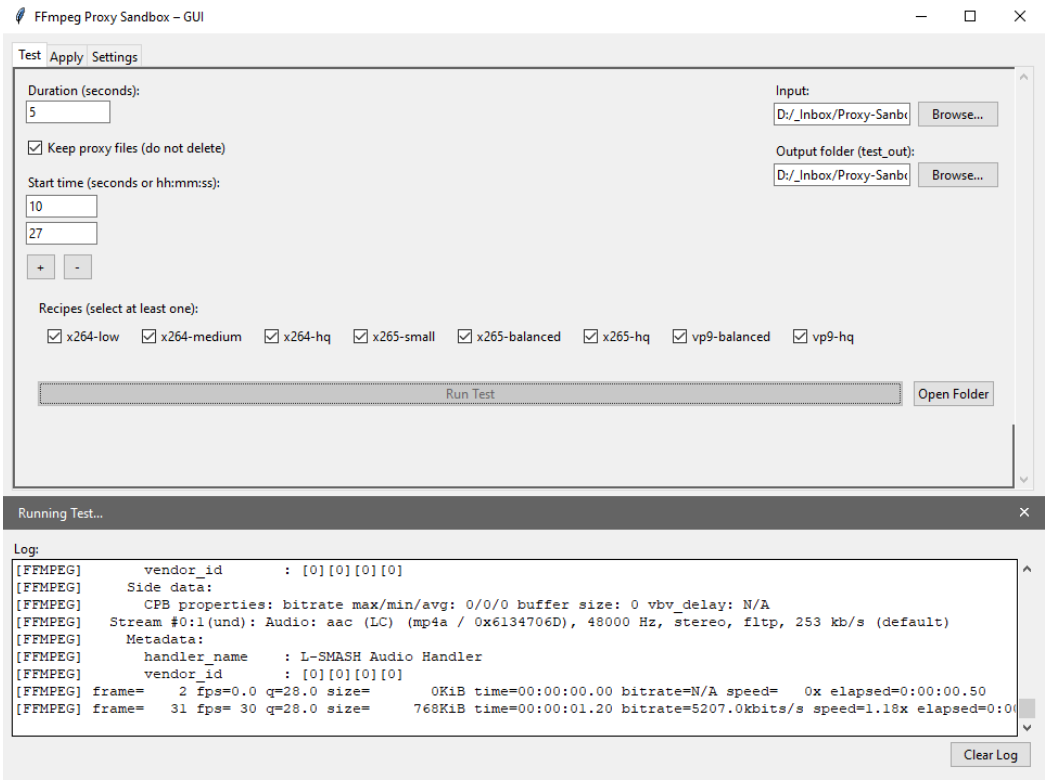
1) membuat proxy clip berdasarkan konfigurasi durasi dan start time (jumlah start time menentukan banyak proxy clip),

2) menjalankan transcoding untuk setiap proxy dan setiap resep yang dipilih,

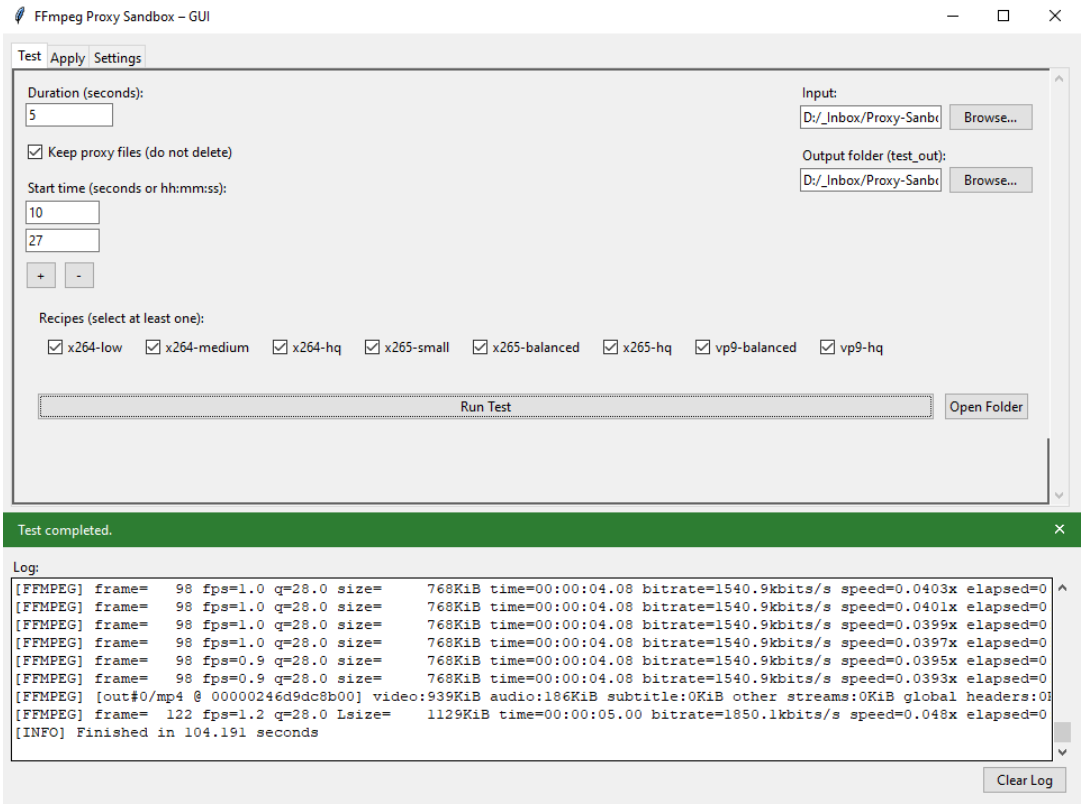
3) menghitung metrik PSNR dan SSIM,

4) menuliskan ringkasan hasil ke berkas summary.csv.

Selama proses berlangsung, area log menampilkan perintah FFmpeg yang dieksekusi beserta progres frame dan bitrate, sehingga pengguna dapat menelusuri detail teknis.

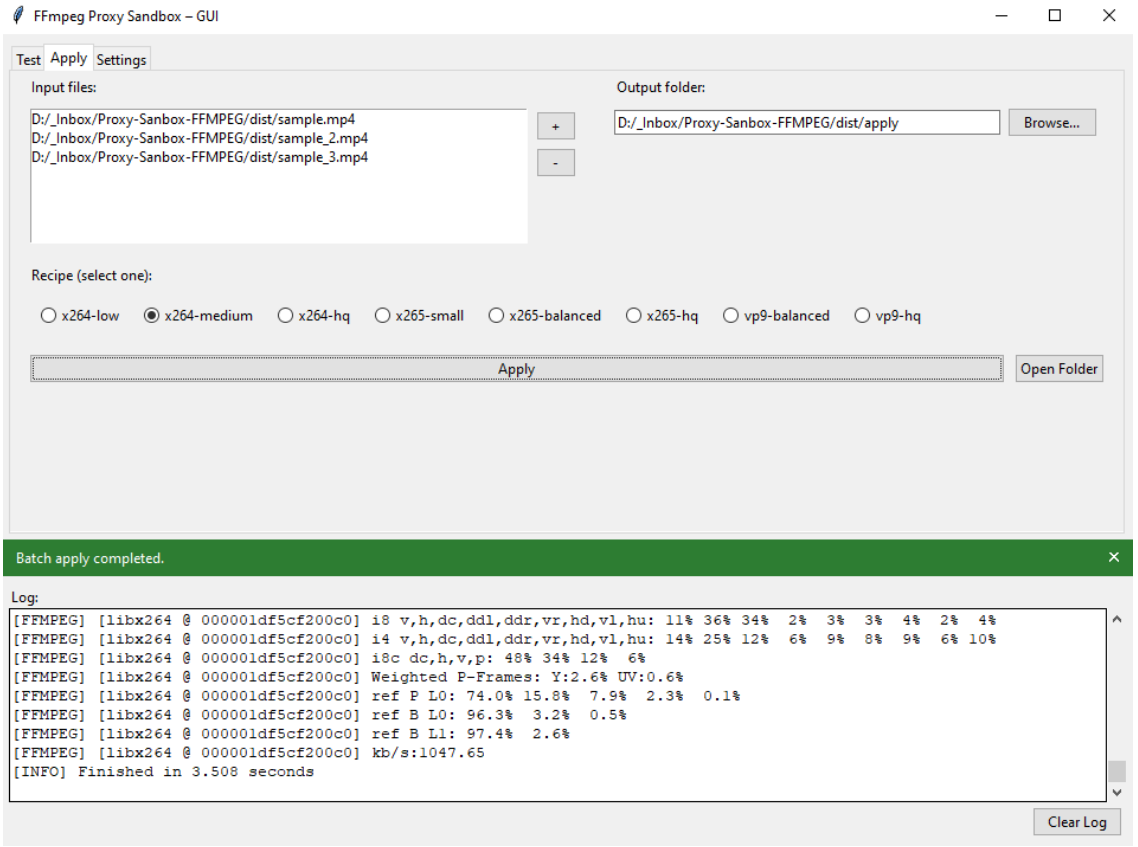



gambar 1.2 screenshot tab Test saat proses sedang berjalan dengan log FFmpeg muncul



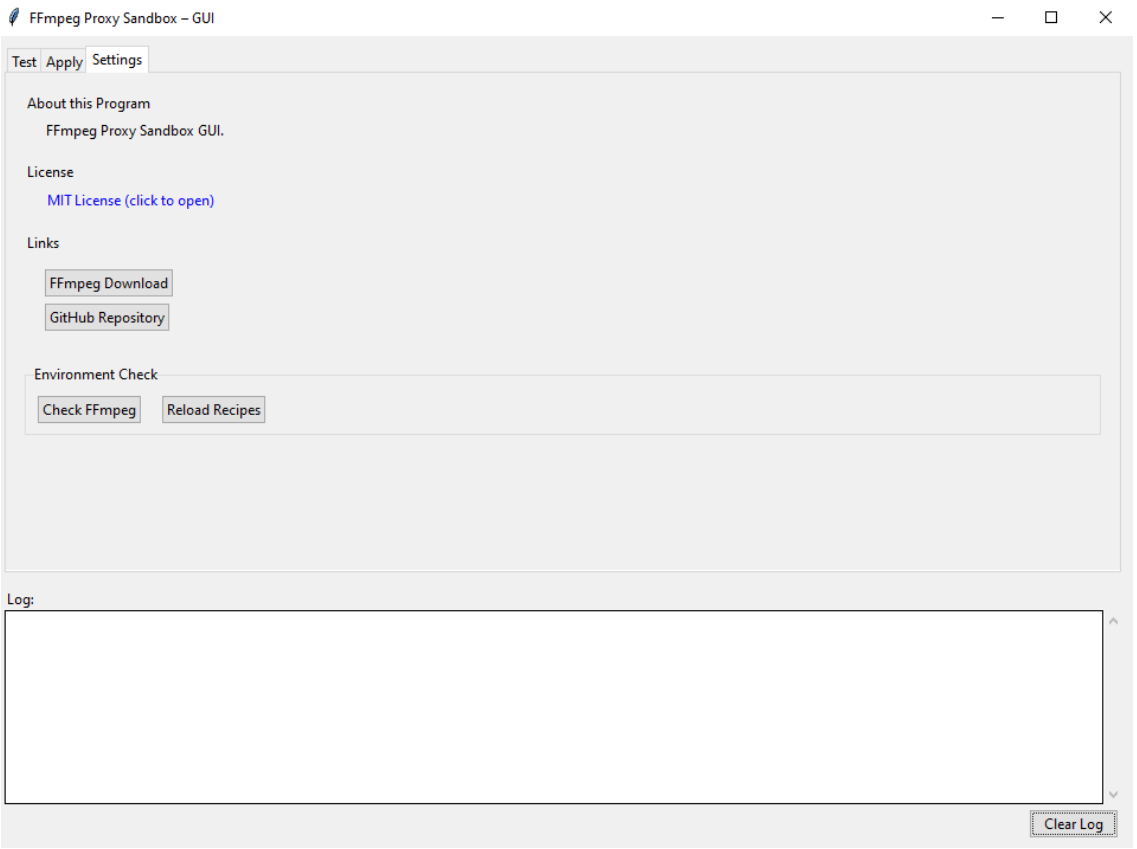
gambar 1.3 screenshot tab Test setelah proses selesai dengan log ringkas dan notifikasi status proses

Tab Apply digunakan untuk menerapkan satu resep transcoding ke satu atau beberapa video penuh. Pengguna menambahkan daftar file input, memilih satu resep (misalnya x264-medium), menentukan folder keluaran, lalu menjalankan proses. Hasilnya berupa video baru dengan penamaan <nama_asli>_<id_resep>.mp4.




gambar 1.4 screenshot tab Apply setelah proses batch apply selesai, beserta log singkat di bagian bawah

Tab Settings memuat informasi program, lisensi MIT, tautan ke dokumentasi FFmpeg dan repositori GitHub, serta tombol untuk mengecek ketersediaan FFmpeg dan memuat ulang berkas recipes.json.




gambar 1.5 screenshot tab Settings (About, License, Check FFmpeg, Reload Recipes)

Tampilan ini menunjukkan bahwa sistem tidak hanya menjalankan pipeline secara otomatis, tetapi juga menjaga transparansi melalui log perintah FFmpeg dan struktur keluaran yang teratur.

### 3.2 Skenario Pengujian

Pengujian dilakukan pada lingkungan:

- CPU: Intel Core i3-12100F
- RAM: 16 GB
- GPU: AMD Radeon RX 570 (4 GB)
- Storage: SSD SATA 256GB

- Sistem operasi: Windows 64-bit

- FFmpeg: *ffmpeg version 2025-10-21-git-535d4047d3-essentials_build-www.gyan.dev*

### 3.2.1 Video uji dan pembuatan proxy

Video utama yang digunakan untuk pengujian proxy adalah:

sample.mp4
Resolusi 1920×1080
Durasi ± 1 menit 7 detik
Ukuran berkas sekitar 38–39 MB

Dari video ini dibuat dua proxy clip dengan konfigurasi:

- durasi proxy 5 detik

- start time 10 detik (Proxy 1) dan 27 detik (Proxy 2)

- opsi keep proxy files diaktifkan

Untuk setiap proxy, sistem menjalankan semua resep pada recipes.json:

- H.264: x264-low, x264-medium, x264-hq

- H.265: x265-small, x265-balanced, x265-hq

- VP9: vp9-balanced, vp9-hq

Setiap kombinasi menghasilkan satu baris di summary.csv yang memuat ukuran proxy, ukuran hasil encoding, waktu proses, PSNR, dan SSIM.

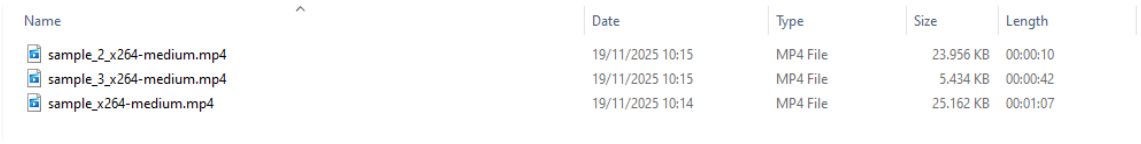
### 3.2.2 Pengujian apply ke video penuh

Sebagai ilustrasi penggunaan Apply, resep x264-medium diterapkan ke tiga video penuh:

1) sample.mp4 – 1920×1080, 00:01:07, ±38,4 MB
2) sample_2.mp4 – 2160×3840, 00:00:10, ±27,0 MB
3) sample_3.mp4 – 848×478, 00:00:42, ±7,7 MB

Hasil apply berupa:

- sample_x264-medium.mp4
- sample_2_x264-medium.mp4
- sample_3_x264-medium.mp4




gambar 1.6 screenshot folder output berisi tiga hasil apply beserta ukuran dan durasi

### 3.3 Hasil Uji Proxy dan Perbandingan Resep

Tabel berikut merangkum hasil pengujian pada Proxy 1 (start 10 detik, durasi 5 detik). Ukuran dalam MB, reduksi ukuran dalam persen terhadap proxy asli.

Tabel 1. Ringkasan hasil transcoding Proxy 1 (durasi 5 detik) untuk delapan resep.

| Resep | Codec | Ukuran proxy (MB) | Ukuran hasil (MB) | Reduksi ukuran (%) | Waktu encode (s) | PSNR (dB) | SSIM |
|---|---|---|---|---|---|---|---|
| x264-low | H.264 | 4,47 | 1,73 | 61 | 1,87 | 43,56 | 0,98 |
| x264-medium | H.264 | 4,47 | 2,74 | 39 | 2,66 | 46,30 | 0,98 |
| x264-hq | H.264 | 4,47 | 4,47 | 0 | 4,57 | 48,68 | 0,99 |
| x265-small | H.265 | 4,47 | 0,87 | 81 | 3,59 | 42,69 | 0,97 |
| x265-balanced | H.265 | 4,47 | 1,15 | 74 | 9,91 | 44,31 | 0,98 |
| x265-hq | H.265 | 4,47 | 1,90 | 58 | 52,30 | 46,35 | 0,98 |
| vp9-balanced | VP9 | 4,47 | 1,71 | 62 | 11,48 | 45,30 | 0,98 |
| vp9-hq | VP9 | 4,47 | 2,20 | 51 | 129,39 | 46,35 | 0,98 |

Secara umum, semua resep menghasilkan kualitas tinggi dengan PSNR 42–49 dB dan SSIM di atas 0,97. Perbedaannya terlihat pada trade-off kualitas, ukuran, dan waktu encode:

- H.264: menawarkan keseimbangan kecepatan dan kualitas, dari kompresi tinggi hingga hampir lossless.
- H.265: memberikan efisiensi kompresi lebih baik, namun waktu encode jauh lebih lama, terutama mode high quality.
- VP9: kualitas kompetitif, tetapi waktu encode paling lama.

Hasil pada Proxy 2 menunjukkan kecenderungan yang sama.

### 3.4 Hasil Apply Resep ke Video Penuh

Tabel berikut merangkum efek apply x264-medium pada tiga video penuh.

gambar 1.8 tabel 2 perbandingan ukuran video penuh sebelum dan sesudah apply x264-medium

Tabel 2. Perbandingan ukuran video penuh sebelum dan sesudah apply x264-medium

| Video | Resolusi | Durasi | Ukuran asli (MB) | Ukuran x264-medium (MB) | Reduksi ukuran (%) |
|---|---|---|---|---|---|
| sample.mp4 | 1920×1080 | 00:01:07 | 38,4 | 24,6 | 36,0 |
| sample_2.mp4 | 2160×3840 | 00:00:10 | 27,0 | 23,4 | 13,3 |
| sample_3.mp4 | 848×478 | 00:00:42 | 7,7 | 5,3 | 31,3 |

Pengamatan:

- Video Full HD menunjukkan reduksi 36%
- Video 4K vertikal menunjukkan reduksi lebih kecil, 13%
- Video SD-ish menunjukkan reduksi 31%

### 3.5 Pembahasan

1) Sistem memenuhi fungsi eksplorasi setelan transcoding.
   Pipeline mampu menghasilkan proxy, menjalankan transcoding, menghitung PSNR–SSIM, dan menulis summary.csv.

2) Trade-off kualitas–ukuran–waktu terlihat jelas.

   a) H.264: cepat, kualitas stabil.

b) H.265: efisien, waktu encode tinggi.

c) VP9: kualitas bagus, namun sangat lambat.

3) Metrik PSNR dan SSIM memudahkan pembacaan pengaruh parameter.
Peningkatan kualitas terlihat dari naiknya PSNR dan SSIM, terutama pada mode high quality.

4) Fitur batch apply menunjukkan kegunaan praktis.
Resep yang dianggap optimal melalui pengujian proxy dapat langsung diterapkan ke banyak video dalam workflow nyata.

Secara keseluruhan, FFmpeg Proxy Sandbox bekerja sesuai rancangan: mempermudah eksperimen transcoding berbasis cuplikan, mengungkap trade-off kualitas–ukuran–waktu antar codec, dan menyajikan hasil dalam bentuk data terstruktur yang siap digunakan dalam analisis dan pelaporan.

## KESIMPULAN

Penelitian ini telah menghasilkan sebuah prototipe alat uji transcoding video berbasis FFmpeg yang mengimplementasikan pendekatan uji cepat menggunakan cuplikan (proxy). Sistem mampu membuat beberapa proxy dari satu video sumber, menjalankan transcoding untuk sejumlah resep codec yang dipilih (H.264, H.265, dan VP9), menghitung metrik kualitas objektif (PSNR dan SSIM), serta merekap ukuran berkas dan waktu proses dalam satu berkas ringkasan berformat CSV. Antarmuka grafis berbasis Tkinter membantu pengguna mengoperasikan pipeline tersebut tanpa harus menulis perintah FFmpeg secara manual, namun tetap menampilkan log perintah sehingga proses tetap transparan dan edukatif.

Hasil pengujian menunjukkan bahwa alat ini dapat mengungkap trade-off kualitas–ukuran–waktu di antara berbagai setelan transcoding. Resep H.265 dan VP9 umumnya memberikan pengurangan ukuran berkas yang lebih besar dibanding H.264 pada kualitas objektif yang masih tinggi, tetapi dengan konsekuensi waktu encoding yang lebih panjang, terutama pada mode kualitas tinggi. Sementara itu, fitur apply batch memungkinkan resep

yang dianggap seimbang—misalnya `x264-medium`—langsung diterapkan pada beberapa video penuh dan memberikan penghematan ukuran berkas yang nyata.

Meskipun demikian, sistem ini masih memiliki keterbatasan, antara lain belum adanya pengukuran kualitas subjektif dari pengguna dan belum adanya mekanisme otomatis untuk membandingkan hasil antar skenario uji dalam bentuk visual interaktif. Pengembangan berikutnya dapat mengarah pada integrasi metrik tambahan, dukungan preset uji yang lebih kompleks, serta evaluasi usability dengan melibatkan pengguna akhir agar sistem ini semakin matang sebagai alat bantu eksperimen dan pembelajaran di bidang sistem multimedia.

## DAFTAR PUSTAKA

[1] Benjak, J., Hofman, D., Knezović, J., & Žagar, M. (2022). Performance Comparison of H.264 and H.265 Encoders in a 4K FPV Drone Piloting System. Applied Sciences, 12(13), 6386. https://doi.org/10.3390/app12136386

[2] Iza Paredes, C., Mohamad-Mezher, A., & Aguilar Igartua, M. (2017). Performance Comparison of H.265/HEVC, H.264/AVC and VP9 Encoders in Video Dissemination over VANETs. In Smart Objects and Technologies for Social Good (GOODTECHS 2016), Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 195, 51–60. https://doi.org/10.1007/978-3-319-61949-1_6

[3] Vranješ, M., Rimac-Drlje, S., & Grgić, K. (2013). Review of Objective Video Quality Metrics and Performance Comparison Using Different Databases. Signal Processing: Image Communication, 28(1), 1–19. https://doi.org/10.1016/j.image.2012.10.003

[4] Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image Quality Assessment: From Error Visibility to Structural Similarity. IEEE Transactions on Image Processing, 13(4), 600–612. https://doi.org/10.1109/TIP.2003.819861

[5] Shahid, M., Rossholm, A., Lövström, B., & Zepernick, H.-J. (2014). No-reference Image and Video Quality Assessment: A Classification and Review of Recent Approaches. EURASIP Journal on Image and Video Processing, 2014(40), 1–32. https://doi.org/10.1186/1687-5281-2014-40

[6] Shi, Y.-Q., & Sun, H. (2019). Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards (3rd ed.). CRC Press. https://doi.org/10.1201/9781315097954

[7] Pressman, R. S., & Maxim, B. R. (2014). Software Engineering: A Practitioner's Approach (8th ed.). McGraw-Hill Education. (Informasi bibliografis dapat diverifikasi melalui katalog penerbit dan indeks sitasi buku)

[8] FFmpeg Project. (diakses 2025). ffmpeg-all Documentation: ffmpeg Tool and Components Overview. https://ffmpeg.org/ffmpeg-all.html