



Perbandingan Algoritma Yamamoto's Recursive Code dan LZW untuk Kompresi File Video

Khairani Resty Harahap^{1*}, Firahmy Rizky²

^{1*,2}Program Studi Sistem Informasi, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Muhammadiyah Sumatera Utara, Medan, Indonesia

Email: ^{1*}khairanirestyhrp@gmail.com, ²firahmirizky@umsu.ac.id

Abstrak

Ukuran file video digital yang besar menimbulkan kebutuhan akan metode kompresi lossless yang andal untuk menghemat ruang penyimpanan tanpa mengurangi kualitas data. Penelitian ini bertujuan mengimplementasikan dan membandingkan kinerja dua algoritma kompresi lossless, yaitu Yamamoto's Recursive Code (YRC) dan LZW (Lempel-Ziv-Welch), dalam sistem kompresi video berbasis web bernama VidCompress yang dibangun menggunakan Python dengan framework Flask. Pengujian dilakukan terhadap 6 sesi kompresi yang mencakup 5 file video MP4/H.264 unik dengan variasi ukuran 0,94–6,20 MB, dengan satu file diuji dua kali sebagai replikasi, menghasilkan 12 data kompresi yang diukur berdasarkan Ratio of Compression (RC), Compression Ratio (CR), Space Saving (SS), dan waktu proses, serta divalidasi melalui pengujian dekompresi langsung terhadap seluruh berkas hasil kompresi. Hasil penelitian menunjukkan bahwa Algoritma YRC secara konsisten memperbesar ukuran file pada seluruh 6 sesi (rata-rata SS –47,18%), namun seluruh hasil dekompresinya (6/6) tervalidasi identik dengan file asli. Algoritma LZW hanya berhasil mengompresi 3 dari 6 sesi (rata-rata SS –2,24%), dan seluruh hasil dekompresinya (6/6) dinyatakan tidak valid akibat keterbatasan implementasi lebar bit kode kamus yang dibatasi 16 bit tanpa mekanisme reset. Keunggulan kedua algoritma bersifat bersyarat: LZW unggul pada rasio kompresi namun belum andal, sedangkan YRC andal namun tidak efisien pada data video MP4/H.264 yang telah terkompresi oleh codec. Perbaikan mekanisme lebar bit LZW direkomendasikan untuk penelitian selanjutnya.

Kata Kunci: kompresi video, lossless compression, Yamamoto's Recursive Code, LZW, Lempel-Ziv-Welch.

PENDAHULUAN

Perkembangan teknologi informasi dan komunikasi telah mendorong meningkatnya volume data digital, khususnya file video yang memiliki ukuran paling besar dan paling kompleks di antara berbagai format data. Sebuah file video Full HD berdurasi satu menit saja dapat mencapai lebih dari 1 GB, dan dengan semakin masifnya penggunaan platform berbagi video, tekanan terhadap kapasitas penyimpanan maupun bandwidth jaringan semakin berat (Irawati et al., 2025). Permasalahan utama dalam pengelolaan file video adalah ketidakseimbangan antara kapasitas penyimpanan yang tersedia dengan ukuran file yang terus membesar. Kondisi ini menjadi hambatan nyata bagi individu, institusi pendidikan, perusahaan media, maupun pengembang sistem informasi.

Kompresi data lossless telah lama menjadi solusi yang menjamin integritas penuh data karena dapat mengembalikan data ke bentuk semula secara sempurna setelah dekompresi (Irawati et al., 2025). Karena tidak ada satu algoritma pun yang secara mutlak unggul dalam semua kondisi dan jenis data, perbandingan antar algoritma menjadi langkah penting sebelum implementasi. Di antara berbagai algoritma lossless yang tersedia, dua algoritma yang memiliki karakteristik komplementer dan belum pernah dibandingkan secara empiris pada data video adalah Yamamoto's Recursive Code (YRC) dan LZW (Lempel-Ziv-Welch).

Beberapa penelitian terdahulu telah mengkaji kinerja kedua algoritma ini secara terpisah. Nasution (2023) membuktikan bahwa Algoritma YRC mampu mengompresi file gambar BMP dengan Compression Ratio (CR) 50%–68%, namun penelitian tersebut terbatas pada file gambar statis dan tidak melibatkan file video. Darma Nasution dan Saputra (2022) membandingkan lima algoritma lossless pada data teks dan menemukan bahwa YRC memberikan performa lebih rendah dengan space saving 9%–16% pada data teks, mengindikasikan bahwa kinerjanya sangat bergantung pada jenis data. Dari sisi Algoritma LZW, Betsaida

Situmorang (2024) berhasil mengimplementasikan LZW pada file audio MP3 berukuran 11,6 MB dengan Compression Ratio sebesar 16%, dan Irawati et al. (2025) menganalisis secara komparatif beberapa algoritma termasuk LZW yang menunjukkan keunggulan pada kompresi data biner dengan rasio 35%–50%. Novantara et al. (2023) menerapkan LZW untuk kompresi citra digital pada aplikasi berbasis web dan membuktikan bahwa algoritma ini efektif untuk data dengan pola berulang. Dalam konteks kompresi file video, Fitrya (2023) membuktikan keberhasilan algoritma Elias Omega Code pada file MP4, dan Rasidi (2023) menunjukkan bahwa Interpolative Coding mampu mengompresi video hingga 47%. Pendekatan berbasis web dalam pemrosesan data digital juga telah terbukti andal pada berbagai aplikasi, termasuk verifikasi integritas data citra secara otomatis (Fauzi & Tanjung, 2026), yang mendukung gagasan bahwa sistem kompresi dan validasi file video pun dapat diimplementasikan secara efisien pada platform web.

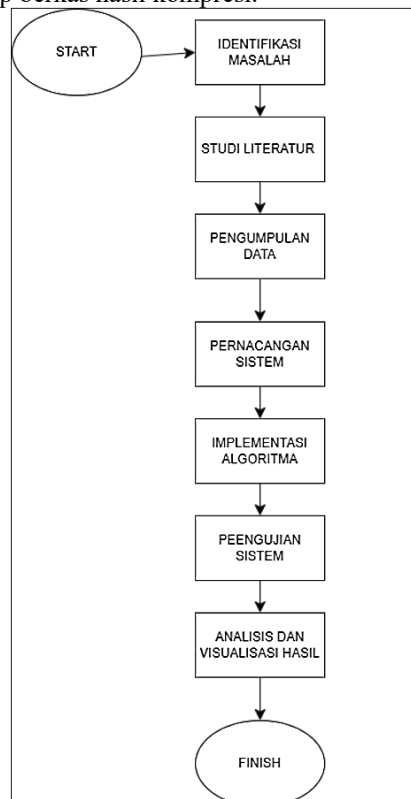
Berdasarkan tinjauan tersebut, terdapat celah penelitian yang signifikan: belum ada penelitian yang secara khusus membandingkan Algoritma YRC dengan LZW dalam konteks kompresi file video. File video memiliki karakteristik jauh lebih kompleks dibanding gambar, teks, maupun audio, karena terdiri dari rangkaian frame berurutan disertai data audio dan metadata terkompresi codec, sehingga pola redundansinya berbeda. Perbandingan kedua algoritma ini perlu dilakukan secara empiris disertai pengujian validasi dekomposisi langsung agar hasilnya dapat menjadi referensi yang sah. Berdasarkan uraian di atas, penelitian ini bertujuan untuk: (1) mengimplementasikan Algoritma YRC dan LZW dalam sistem kompresi file video berbasis web; (2) menganalisis dan membandingkan kinerja kedua algoritma berdasarkan parameter RC, CR, dan SS; serta (3) mengevaluasi keandalan kedua algoritma melalui pengujian dekomposisi dan validasi byte-level pada seluruh berkas hasil kompresi.

METODE

Penelitian ini menggunakan desain eksperimental komparatif dengan pendekatan kuantitatif untuk membandingkan kinerja Algoritma YRC dan LZW pada kompresi file video. Implementasi dilakukan pada sistem kompresi berbasis web bernama VidCompress, dibangun dengan bahasa pemrograman Python menggunakan framework Flask sebagai backend REST API. Frontend dikembangkan dengan HTML, CSS, dan JavaScript, dan komunikasi antara keduanya dilakukan melalui protokol HTTP. Proses kompresi dan dekomposisi bekerja pada level byte stream dari file video, tidak pada level piksel atau frame.

Arsitektur Sistem

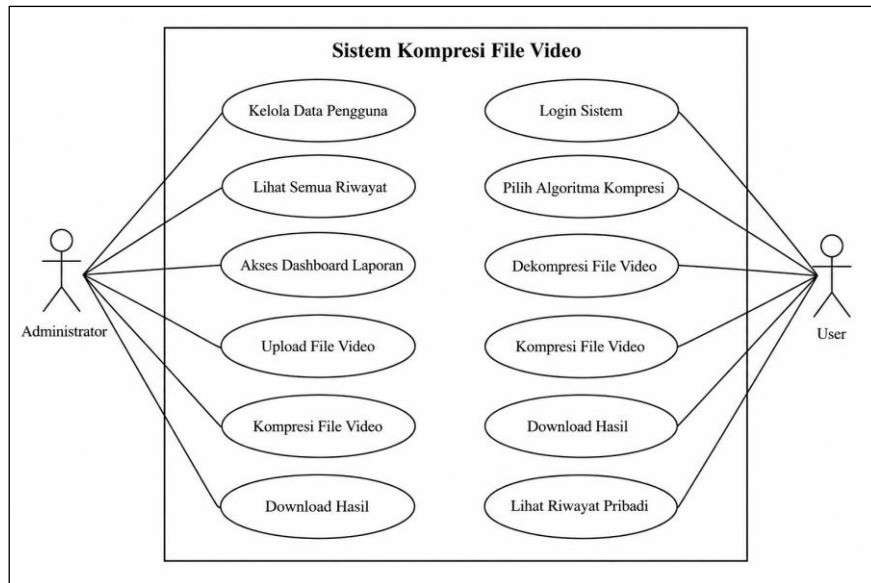
Arsitektur penelitian terdiri atas tahapan: pengumpulan data, implementasi algoritma, pengujian kompresi, pengujian dekomposisi dan validasi, serta analisis perbandingan parameter. Sistem VidCompress menyediakan antarmuka unggah file, pilihan algoritma, tampilan hasil pengujian, serta fitur dekomposisi dan validasi otomatis untuk setiap berkas hasil kompresi.



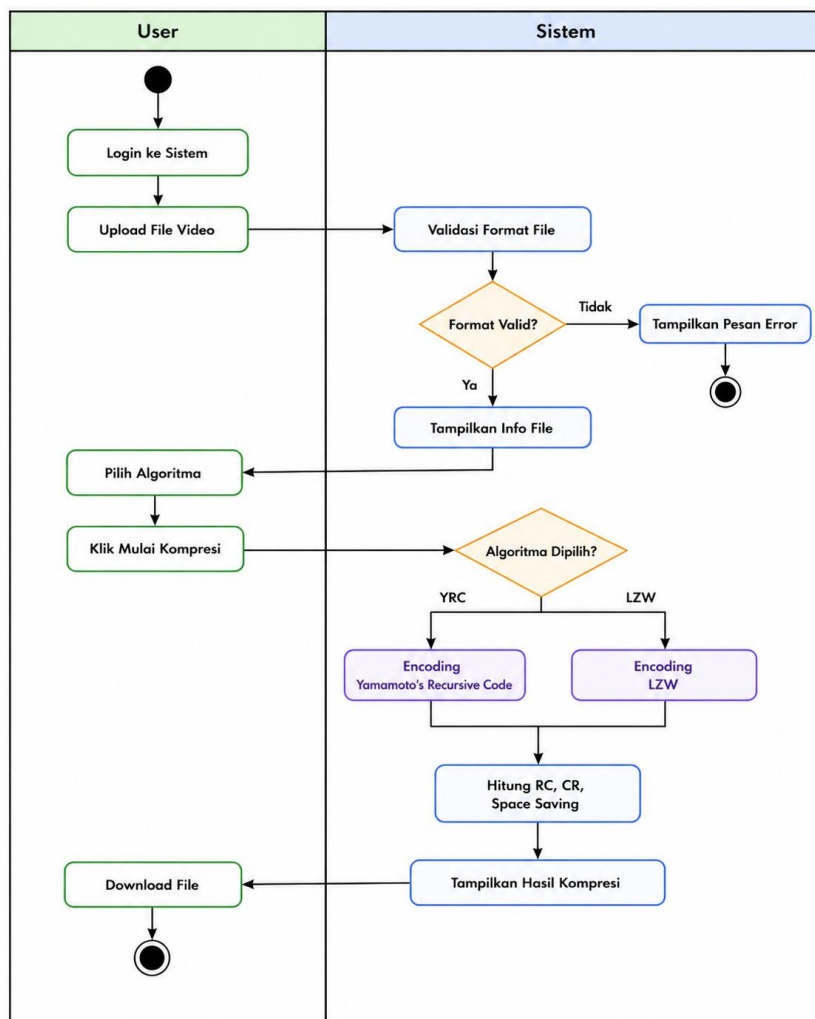
Gambar 1. Arsitektur Penelitian VidCompress

Perancangan Sistem

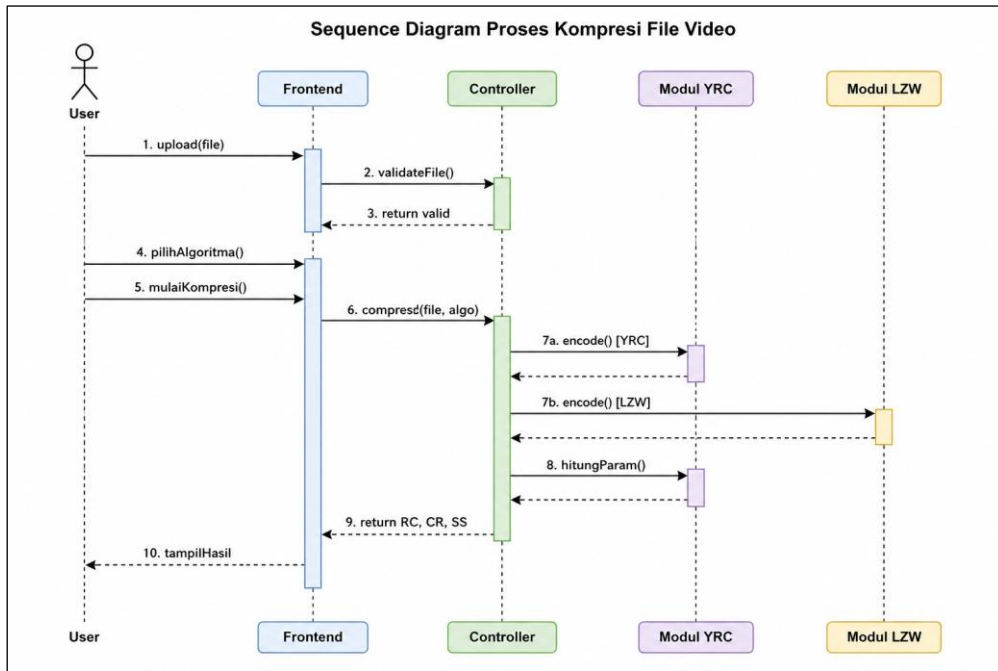
Sistem dirancang dengan dua peran pengguna: administrator yang mengelola data pengguna dan melihat riwayat seluruh sesi, serta user yang mengunggah file, memilih algoritma, dan mengunduh hasil. Diagram use case, activity diagram, dan sequence diagram menggambarkan alur interaksi sistem secara lengkap.



Gambar 2. Use Case Diagram Sistem VidCompress



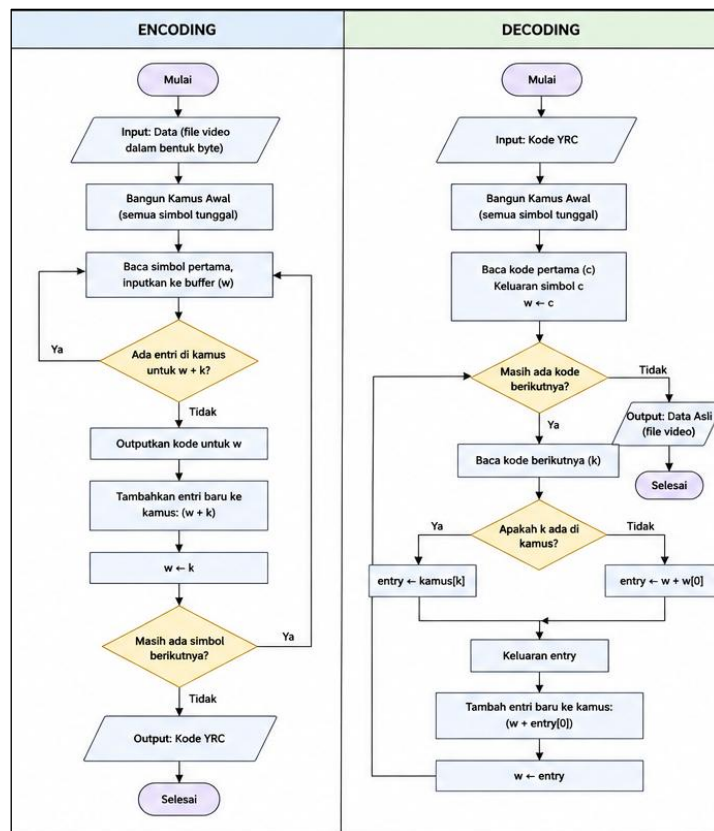
Gambar 3. Activity Diagram Proses Kompresi dan Dekompresi



Gambar 4. Sequence Diagram Kompresi File Video

Algoritma Yamamoto's Recursive Code

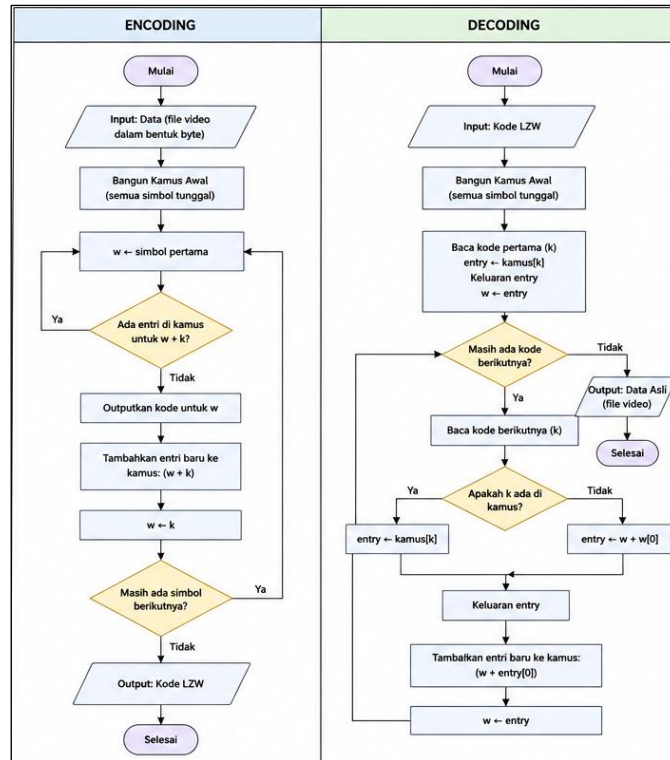
Yamamoto's Recursive Code adalah algoritma kode universal lossless berbasis prinsip rekursif yang dikembangkan oleh Hirosuke Yamamoto (2000). Algoritma ini mengkodekan setiap kode kamus dengan representasi rekursif biner yang dilengkapi delimiter, menghasilkan kode self-delimiting yang memungkinkan dekompresi deterministik tanpa informasi tambahan. Proses encoding dimulai dari representasi biner bilangan positif n , kemudian panjang representasi tersebut dikodekan secara rekursif hingga mencapai kondisi terminasi. Proses decoding merupakan kebalikannya: decoder membaca delimiter dan panjang kode pada setiap level rekursi untuk memulihkan nilai asli secara sempurna (Yamamoto, 2000).



Gambar 5. Flowchart Encoding dan Decoding Yamamoto's Recursive Code

Algoritma LZW (Lempel-Ziv-Welch)

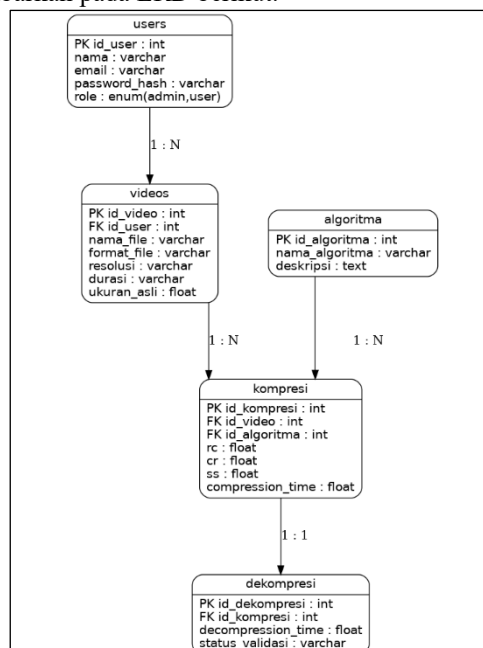
LZW adalah algoritma kompresi adaptif berbasis kamus yang pertama kali dikembangkan oleh Lempel dan Ziv (1977, 1978) dan disempurnakan oleh Welch (1984). Kamus diinisialisasi dengan 256 entri byte tunggal dan diperluas secara dinamis selama kompresi berlangsung. Setiap string berulang dalam data digantikan dengan kode numerik yang lebih pendek, dan kamus dibangun secara identik oleh encoder maupun decoder sehingga tidak perlu disimpan dalam file terkompresi (Welch, 1984). Pada implementasi penelitian ini, lebar bit kode diatur secara adaptif dengan batas maksimum 16 bit ($\text{MAX_BITS} = 16$) tanpa mekanisme reset kamus.



Gambar 6. Flowchart Encoding dan Decoding LZW

Entity Relationship Diagram

Basis data sistem dirancang dengan beberapa entitas utama: users, videos, algoritma, kompresi, dekompresi, api_logs, dan riwayat_aktivitas. Relasi antar entitas terdiri dari one-to-many (1:N) dan one-to-one (1:1) sebagaimana digambarkan pada ERD berikut.



Gambar 7. ERD Sistem Kompresi File Video

Data Pengujian

Lima file video MP4/H.264 unik digunakan sebagai sampel pengujian, dengan videoplayback.mp4 diuji dua kali sebagai replikasi sehingga total terdapat 6 sesi kompresi. Seluruh file berformat MP4 dengan codec H.264, namun dengan variasi resolusi (640×360 hingga 1280×720), frame rate (24–60 fps), orientasi (potret dan lanskap), bitrate (±287 kbps – ±3,03 Mbps), dan durasi (2,57–134,80 detik).

Tabel 1. Data Sampel Pengujian

No.	Nama File	Ukuran	Resolusi	Durasi	FPS	Bitrate
1	videoplayback.mp4 (×2 replikasi)	6,20 MB	640×360	134,80 s	24	±287 kbps
2	WhatsApp_Video_2026-07-22 at 17.18.06.mp4	4,37 MB	1280×720	12,83 s	30	±2,80 Mbps
3	WhatsApp_Video_2026-08-03 at 17.31.54.mp4	2,51 MB	464×832	10,31 s	60	±1,93 Mbps
4	WhatsApp_Video_2026-08-03 at 17.32.07.mp4	0,94 MB	1280×720	2,57 s	30	±3,03 Mbps
5	WhatsApp_Video_2026-08-03 at 17.32.08.mp4	1,07 MB	1280×720	2,93 s	30	±3,01 Mbps

Parameter Pengujian

Parameter kinerja kompresi yang digunakan dalam penelitian ini didefinisikan sebagai berikut, di mana S_n adalah ukuran asli (original size) dan S^d adalah ukuran hasil kompresi (compressed size):

$$RC = S^d / S_n \quad (1)$$

$$CR = S_n / S^d \quad (2)$$

$$SS = (1 - RC) \times 100\% \quad (3)$$

Nilai RC mendekati 0 atau CR jauh di atas 1 menandakan kompresi yang baik. Nilai $RC > 1$ atau $CR < 1$ menandakan ekspansi ukuran. Seluruh berkas hasil kompresi (12 berkas dari 6 sesi × 2 algoritma) diuji validitas dekompresinya dengan cara membandingkan byte hasil dekompresi dengan file asli secara langsung melalui sistem.

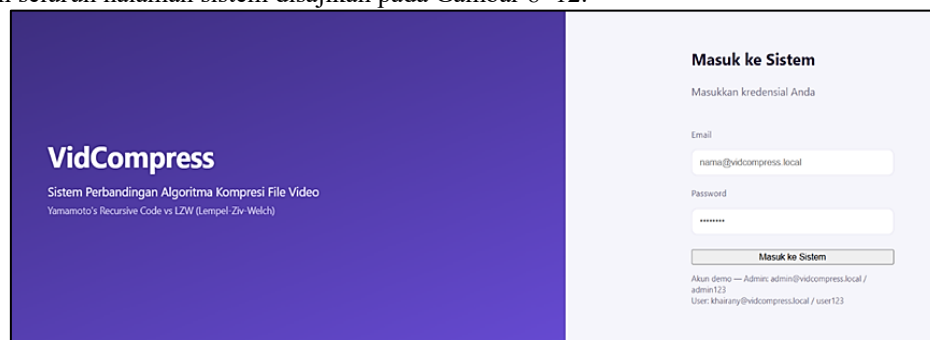
Tabel 2. Parameter Pengujian Algoritma

Parameter	Rumus	Interpretasi
RC (Ratio of Compression)	$RC = S^d / S_n$	Nilai mendekati 0 = kompresi terbaik
CR (Compression Ratio)	$CR = S_n / S^d$	Nilai > 1 = berhasil dikompres
SS (Space Saving)	$SS = (1 - RC) \times 100\%$	Nilai mendekati 100% = penghematan terbesar

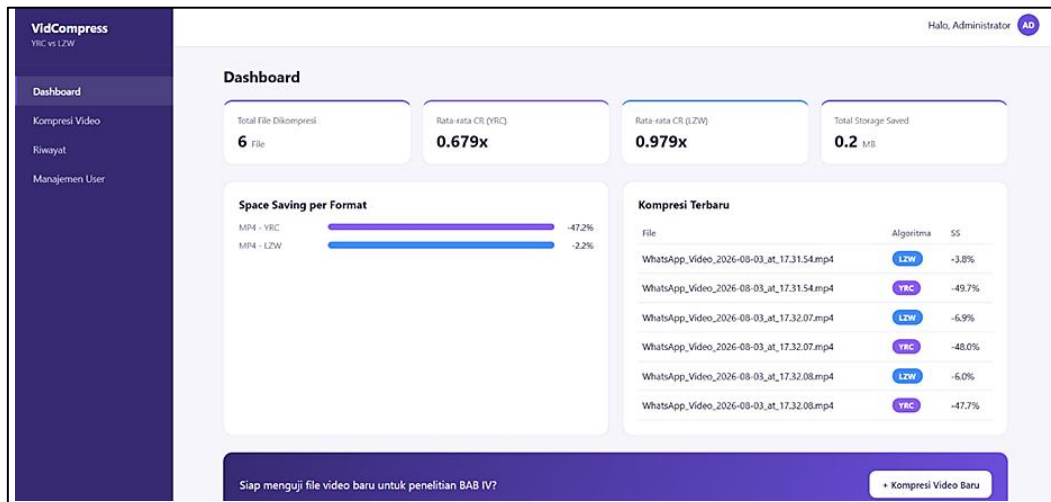
HASIL DAN PEMBAHASAN

Implementasi Sistem VidCompress

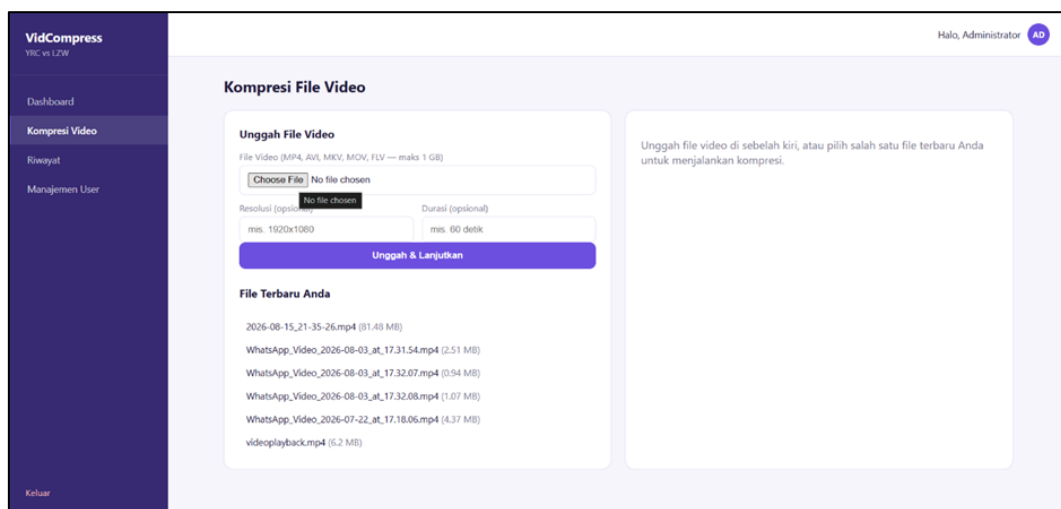
Sistem VidCompress berhasil diimplementasikan sesuai rancangan pada Bab Metode. Halaman login menyediakan autentikasi dua peran dengan kata sandi tersimpan dalam bentuk hash menggunakan `werkzeug.security`. Dashboard menampilkan ringkasan pengujian secara real-time yang dihitung dari seluruh data di tabel kompresi, mencakup total file dikompresi, rata-rata CR per algoritma, dan rata-rata SS per format. Halaman kompresi memungkinkan pengguna mengunggah file video, memilih algoritma, menjalankan kompresi, dan langsung melakukan dekompresi serta validasi dalam satu antarmuka. Halaman riwayat menyajikan seluruh rekam jejak sesi pengujian yang menjadi sumber utama data pada Tabel 3–5. Tampilan seluruh halaman sistem disajikan pada Gambar 8–12.



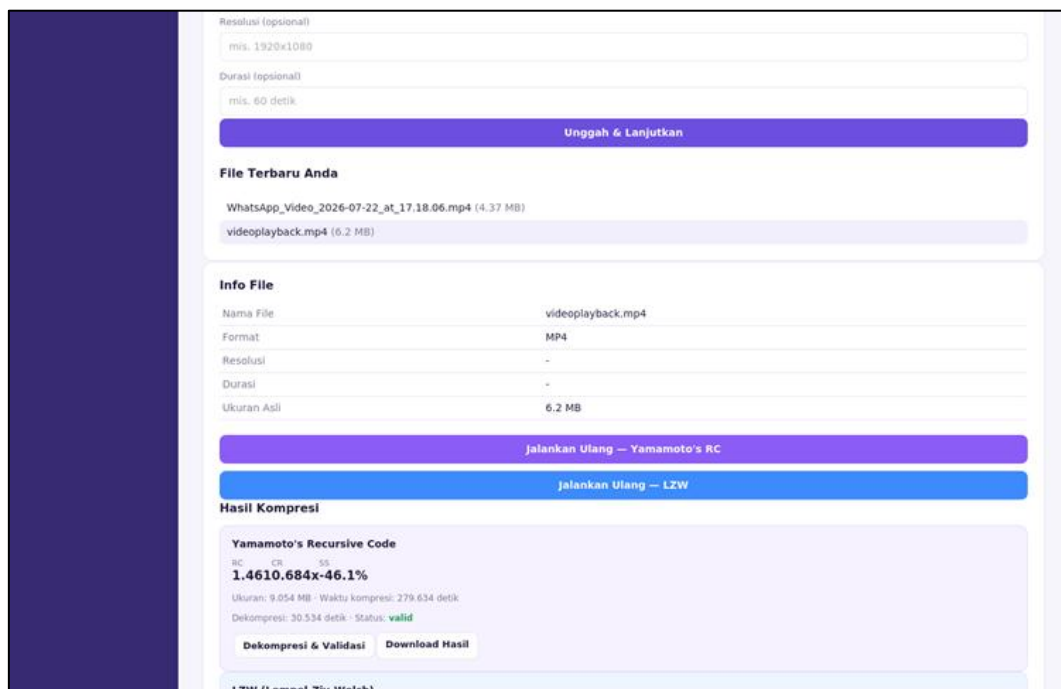
Gambar 8. Tampilan Halaman Login



Gambar 9. Tampilan Halaman Dashboard



Gambar 10. Tampilan Halaman Unggah Video



Gambar 11. Tampilan Halaman Hasil Kompresi, Dekompresi, dan Validasi

Gambar 12. Tampilan Halaman Riwayat Kompresi dan Manajemen Pengguna

Verifikasi Perhitungan Manual Algoritma

Untuk memverifikasi kebenaran logika program secara eksplisit, disajikan contoh perhitungan manual menggunakan string sederhana "ABABABABA" (9 byte). Kamus awal berisi 256 simbol byte tunggal. Proses pembentukan kamus identik di kedua algoritma, sebagaimana ditunjukkan pada Tabel 3.

Tabel 3. Tahap Pembentukan Kamus untuk Input "ABABABABA"

Langkah	w (awalan dikenal)	Kode w	wc (kombinasi baru)	Kode kamus baru
1	A	65	AB	256
2	B	66	BA	257
3	AB	256	ABA	258
4	ABA	258	ABAB	259
5	BA	257	(akhir data)	—

Deretan 5 kode kamus yang dihasilkan: 65, 66, 256, 258, 257. Tabel 4 menyajikan representasi bit untuk skema YRC dan LZW pada setiap kode tersebut.

Tabel 4. Representasi Bit YRC vs LZW untuk Setiap Kode Kamus

Kode Kamus	n = kode+1	Biner(n)	Kode Rekursif YRC	Bit YRC	Bit LZW
65	66	1000010	1011010000100	13 bit	9 bit
66	67	1000011	1011010000110	13 bit	9 bit
256	257	100000001	1110001000000010	16 bit	9 bit
258	259	100000011	1110001000000110	16 bit	9 bit
257	258	100000010	1110001000000100	16 bit	9 bit
Total				74 bit → 11 byte	45 bit → 7 byte

Hasil manual: ukuran akhir YRC = 11 byte dan LZW = 7 byte untuk input 9 byte, identik dengan keluaran fungsi compress() pada sistem. Kedua algoritma berhasil merekonstruksi data secara sempurna (round-trip valid) pada data uji sederhana ini, memverifikasi kebenaran logika program sebelum pengujian pada data video sesungguhnya.

Hasil Pengujian Kompresi Algoritma Yamamoto's Recursive Code

Pengujian dilakukan dengan mengompresi keenam sesi video menggunakan Algoritma YRC. Tabel 5 menyajikan hasil pengujian yang diambil langsung dari halaman Riwayat Kompresi sistem.

Tabel 5. Hasil Pengujian Kompresi Algoritma Yamamoto's Recursive Code

No	Nama File	Ukuran	RC	CR	SS (%)	T. Kompresi	T. Dekompresi	Validasi
1	WhatsApp_...17.31.54.mp4	2,51 MB	1,4969	0,6680x	-49,69%	141,64 s	11,16 s	Valid
2	WhatsApp_...17.32.07.mp4	0,94 MB	1,4795	0,6759x	-47,95%	31,46 s	3,75 s	Valid

No	Nama File	Ukuran	RC	CR	SS (%)	T. Kompresi	T. Dekompresi	Validasi
3	WhatsApp_...17.32.08.mp4	1,07 MB	1,4773	0,6769x	-47,73%	33,85 s	4,74 s	Valid
4	videoplayback.mp4 (run-1)	6,20 MB	1,4613	0,6843x	-46,13%	279,63 s	31,23 s	Valid
5	WhatsApp_...17.18.06.mp4	4,37 MB	1,4546	0,6875x	-45,46%	411,69 s	19,35 s	Valid
6	videoplayback.mp4 (run-2)	6,20 MB	1,4613	0,6843x	-46,13%	164,72 s	28,82 s	Valid
Rata-rata			1,4718	0,6795x	-47,18 %	177,17 s	16,51 s	6/6 (100%)

Berdasarkan Tabel 5, diperoleh nilai rata-rata RC sebesar 1,4718 dengan CR 0,6795x. Karena RC > 1 pada seluruh 6 sesi, Algoritma YRC secara konsisten memperbesar ukuran file, dengan rata-rata Space Saving -47,18%. Rentang SS yang sempit (-45,46% hingga -49,69%, selisih 4,23 poin persentase) meskipun ukuran dan karakteristik video bervariasi signifikan, mengonfirmasi bahwa fenomena ekspansi bersifat sistemik terhadap kombinasi algoritma dan jenis data (video MP4/H.264). Replikasi pengujian pada videoplayback.mp4 menghasilkan RC, CR, dan SS yang identik persis (1,4613/0,6843x/-46,13%) pada kedua run, membuktikan sifat deterministik algoritma. Variasi pada waktu kompresi (279,63 s vs 164,72 s) dipengaruhi oleh beban sistem, bukan logika algoritma. Dari sisi keandalan, seluruh 6 hasil dekompresi tervalidasi identik byte demi byte dengan file asli (6/6 valid, 100%), konsisten dengan sifat kode rekursif yang self-delimiting.

Hasil Pengujian Kompresi Algoritma LZW

Tabel 6. Hasil Pengujian Kompresi Algoritma LZW

No	Nama File	Ukuran	RC	CR	SS (%)	T. Kompresi	T. Dekompresi	Validasi
1	WhatsApp_...17.31.54.mp4	2,51 MB	1,0377	0,9637x	-3,77 %	129,11 s	5,92 s	Tidak Valid
2	WhatsApp_...17.32.07.mp4	0,94 MB	1,0688	0,9356x	-6,88 %	75,22 s	2,09 s	Tidak Valid
3	WhatsApp_...17.32.08.mp4	1,07 MB	1,0604	0,9430x	-6,04 %	57,62 s	2,47 s	Tidak Valid
4	videoplayback.mp4 (run-1)	6,20 MB	0,9852	1,0150x	+1,48 %	225,10 s	15,01 s	Tidak Valid
5	WhatsApp_...17.18.06.mp4	4,37 MB	0,9972	1,0028x	+0,28 %	272,54 s	11,51 s	Tidak Valid
6	videoplayback.mp4 (run-2)	6,20 MB	0,9852	1,0150x	+1,48 %	112,66 s	14,81 s	Tidak Valid
Rata-rata			1,0224	0,9792x	-2,24 %	145,38 s	8,63 s	0/6 (0%)

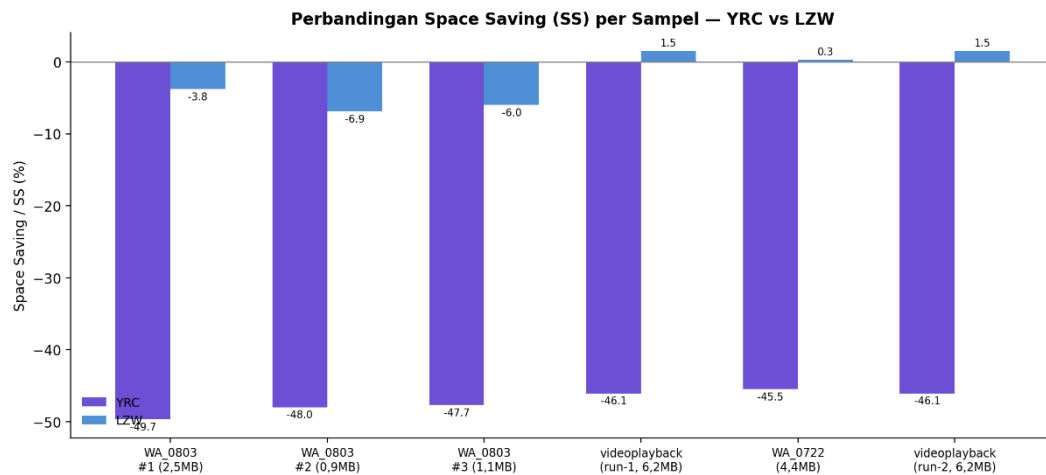
Rata-rata RC LZW sebesar 1,0224 (CR 0,9792x, SS -2,24%) berada pada sisi yang tidak menguntungkan: LZW hanya berhasil mengompresi 3 dari 6 sesi (50%), yaitu pada video dengan bitrate lebih rendah (videoplayback.mp4, ±287 kbps dan WhatsApp_2026-07-22, ±2,80 Mbps), sementara mengalami ekspansi pada tiga video berkecepatan tinggi (±1,93–3,03 Mbps). Temuan ini konsisten dengan keterbatasan algoritma berbasis kamus pada data entropi tinggi: byte stream video terkompresi H.264 memiliki sedikit pola berulang yang dapat dieksploitasi LZW, ditambah overhead penulisan kode kamus (minimum 9 bit per simbol vs 8 bit byte asli) ketika kamus tidak terisi efektif.

Temuan terpenting adalah seluruh 6 hasil dekompresi LZW (100%) dinyatakan tidak valid, termasuk pada video terkecil (0,94 MB). Penyebabnya adalah fungsi `_code_width` pada modul `lzw.py` membatasi lebar kode maksimum pada 16 bit (`MAX_BITS = 16`) tanpa mekanisme reset kamus. Ambang 65.536 entri kamus (2^{16}) tercapai dengan cepat bahkan pada berkas di bawah 1 MB karena entropi byte video H.264 yang tinggi, menyebabkan encoder dan decoder kehilangan sinkronisasi kamus setelah titik tersebut.

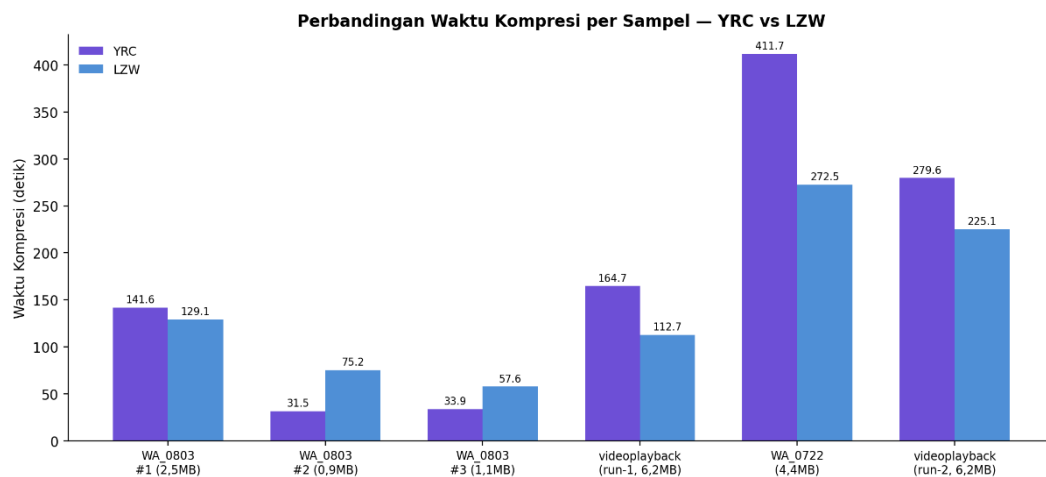
Perbandingan Kinerja Kedua Algoritma

Tabel 7. Perbandingan Hasil Pengujian Algoritma YRC dan LZW

Parameter	YRC	LZW	Keterangan
RC (rata-rata, 6 sesi)	1,4718	1,0224	Nilai mendekati 0 = terbaik → LZW lebih baik, namun keduanya > 1
CR (rata-rata, 6 sesi)	0,6795x	0,9792x	Nilai > 1 = berhasil dikompres → keduanya rata-rata < 1
SS (rata-rata, 6 sesi)	-47,18%	-2,24%	YRC ekspansi 6/6; LZW ekspansi 3/6, kompresi berhasil 3/6
Waktu Kompresi (rata-rata)	177,17 s	145,38 s	LZW lebih cepat pada 4/6 sesi; YRC lebih cepat pada 2/6 sesi
Waktu Dekompresi (rata-rata)	16,51 s	8,63 s	LZW lebih cepat, namun seluruh hasil dekompresinya tidak valid
Validitas Dekompresi (6 sesi, diuji langsung)	6/6 valid (100%)	0/6 valid (0%)	YRC unggul mutlak dari sisi keandalan



Gambar 13. Grafik Perbandingan Space Saving (SS) per Sampel Video



Gambar 14. Grafik Perbandingan Waktu Kompresi per Sampel Video

Pembahasan

Berdasarkan hasil pengujian pada Tabel 5–7 serta visualisasi pada Gambar 13–14, dapat dikemukakan beberapa temuan utama. Pertama, kedua algoritma berhasil diimplementasikan secara end-to-end, dan kebenaran logikanya telah diverifikasi melalui perhitungan manual (Tabel 3–4) maupun pengujian langsung pada lima video riil, termasuk replikasi yang membuktikan sifat deterministik kedua algoritma. Kedua, baik YRC maupun LZW tidak mencapai tingkat kompresi yang signifikan pada data video MP4/H.264 yang telah melalui proses encoding codec. Hal ini disebabkan codec H.264 sendiri telah

mengeksploitasi sebagian besar redundansi spasial dan temporal dalam video sehingga byte stream yang dihasilkan memiliki entropi tinggi, menyisakan sedikit peluang bagi algoritma kompresi generik level byte (Wiegand et al., 2003).

Ketiga, dari sisi rasio kompresi, LZW lebih unggul daripada YRC pada semua parameter (RC, CR, SS), namun keunggulan ini tidak konsisten: LZW hanya berhasil pada 50% sampel dan gagal total pada tiga video bertbitrate tinggi. Penyebab ekspansi pada YRC lebih sistemik: representasi rekursif dengan delimiter pada setiap level membutuhkan lebih banyak bit dibanding skema lebar tetap LZW ketika kode kamus berada pada rentang indeks yang masih relatif kecil, sebagaimana dibuktikan secara eksplisit pada Tabel 4 (74 bit YRC vs 45 bit LZW untuk string "ABABABABA"). Keempat, dan terpenting, pengujian dekompresi yang dijalankan langsung terhadap seluruh 12 berkas hasil kompresi mengungkap trade-off yang sangat tajam: LZW gagal total (0/6 valid) akibat bug MAX_BITS tanpa reset, sementara YRC berhasil total (6/6 valid). Mengingat keandalan dekompresi adalah syarat mutlak dalam penerapan praktis kompresi lossless, implementasi LZW pada sistem ini belum layak digunakan secara operasional meskipun rasio kompresinya lebih baik.

Hasil penelitian ini memperluas dan melengkapi temuan penelitian terdahulu. Jika Nasution (2023) hanya menguji YRC pada file gambar statis BMP dan mendapatkan CR 50%–68%, penelitian ini menunjukkan bahwa pada data video terkompresi H.264 yang jauh lebih kompleks, YRC konsisten memperbesar ukuran—namun dengan validasi dekompresi 100% yang membuktikan bahwa ekspansi bukan disebabkan kegagalan proses. Darma Nasution dan Saputra (2022) yang menemukan kinerja YRC lebih rendah pada data teks (9%–16% space saving) juga dikuatkan oleh penelitian ini yang menunjukkan pola serupa pada data video. Untuk LZW, hasil penelitian ini melengkapi Betsaida Situmorang (2024) dan Novantara et al. (2023) dengan menunjukkan bahwa keberhasilan LZW sangat bergantung pada karakteristik redundansi data: efektif pada audio dan citra dengan pola berulang, namun tidak konsisten pada video bertitik entropi tinggi, ditambah temuan bug lebar-bit yang konsisten muncul bahkan pada berkas <1 MB.

KESIMPULAN

Kedua algoritma kompresi lossless, YRC dan LZW, berhasil diimplementasikan secara end-to-end pada sistem kompresi file video berbasis web VidCompress menggunakan Python dan Flask. Kebenaran logika program telah diverifikasi melalui perhitungan manual dan pengujian langsung pada lima video riil, termasuk replikasi yang membuktikan sifat deterministik algoritma. Algoritma YRC secara konsisten memperbesar ukuran file pada seluruh 6 sesi pengujian (rata-rata SS -47,18%) namun menghasilkan dekompresi yang tervalidasi 100% (6/6 valid), sejalan dengan sifat kode rekursif yang self-delimiting. Algoritma LZW menunjukkan kinerja rasio kompresi yang lebih baik (rata-rata SS -2,24%) dengan keberhasilan kompresi pada 3 dari 6 sesi, namun seluruh hasil dekompresinya (6/6) dinyatakan tidak valid akibat pembatasan lebar bit kode kamus maksimum 16 bit tanpa mekanisme reset kamus. Kedua algoritma secara keseluruhan tidak berhasil mencapai kompresi yang signifikan pada data video MP4/H.264 yang telah terkompresi oleh codec, karena bekerja pada level byte generik tanpa memanfaatkan redundansi spesifik video. Keunggulan antara kedua algoritma bersifat bersyarat: LZW lebih unggul pada parameter rasio kompresi dan kecepatan proses, sedangkan YRC unggul secara mutlak pada keandalan dan konsistensi hasil dekompresi. Mengingat keandalan dekompresi merupakan syarat mutlak dalam penerapan kompresi lossless yang praktis, implementasi LZW belum layak digunakan secara operasional pada penelitian ini sampai keterbatasan lebar bit diperbaiki. Untuk penelitian selanjutnya disarankan: (1) memperbaiki implementasi LZW dengan menambahkan mekanisme reset atau perluasan batas MAX_BITS agar dekompresi dapat valid; dan (2) melakukan pengujian lanjutan pada video berukuran lebih besar (>10 MB) dengan variasi format container (AVI, MKV) untuk mengamati skalabilitas kedua algoritma secara lebih komprehensif.

DAFTAR PUSTAKA

- Betsaida Situmorang, T. (2024). Perancangan Aplikasi Kompresi File MP3 Dengan Menggunakan Algoritma Lempel Ziv Welch (LZW). *Jurnal Sistem Informasi, Teknik Informatika Dan Teknologi Pendidikan*, 3(2), 60–70. <https://doi.org/10.55338/justikpen.v3i2.87>
- Darma Nasution, S., & Saputra, I. (2022). Performance Analysis of Lossless Type of Compression Algorithm in Compression Data. *International Journal of Information System & Technology Akreditasi*, 6(158), 296–302.
- Fauzi, M. D., & Tanjung, M. A. P. (2026). Deteksi Manipulasi Citra Bukti Struk Pembayaran Uang Kuliah Menggunakan Metode Error Level Analysis dan Histogram Color Shift Berbasis Web. *Indonesian Journal of Multidisciplinary Scientific Studies*, 4(6), 431–439. <https://doi.org/10.33151/ijomss.v4i6.1130>

- Fitrya, S. A. (2023). Perbandingan Algoritma Elias Omega Code Dan Elias Delta Code Dalam Mengkompresi File Video (Mp4). *Bulletin of Information System Research*, 1(3), 110–119. <https://doi.org/10.62866/bios.v1i3.29>
- Irawati, O., Nurhasanah, N., Lumbantoruan, A. T., & Zahrani, S. F. (2025). Analisis Peforma Algoritma Kompresi Huffman, LZW, BZIP2, Zstandard dan Brotli dalam Efisiensi Penyimpanan Data dan Transfer Data. *RIGGS: Journal of Artificial Intelligence and Digital Business*, 4(2), 5163–5169. <https://doi.org/10.31004/riggs.v4i2.1396>
- Nasution, L. N. (2023). Implementasi Algoritma Yamamoto's Recursive Code Dalam Mengkompresi File Gambar. *Bulletin of Artificial Intelligence*, 2(2), 112–117. <https://doi.org/10.62866/buai.v2i2.105>
- Novantara, P., Sugiharto, T., & Priantama, R. (2023). Rancang Bangun Aplikasi Monitoring Kegiatan MBKM & Prestasi Mahasiswa Menggunakan Algoritma Lempel ZIV Welch (LZW) untuk Kompresi Citra Digital. *Jurnal Teknik Industri Terintegrasi*, 6(1), 331–339. <https://doi.org/10.31004/jutin.v6i1.16806>
- Rasidi, M. (2023). Perbandingan Algoritma Stout Code Dan Punctured Elias Code Dalam Mengkompresi File Matroska Video Container. *Bulletin of Information System Research*, 2(1), 10–20. <https://doi.org/10.62866/bios.v2i1.88>
- Sayood, K. (2017). *Introduction to data compression* (5th ed.). Morgan Kaufmann.
- Welch, T. A. (1984). A technique for high-performance data compression. *Computer*, 17(6), 8–19. <https://doi.org/10.1109/MC.1984.1659158>
- Wiegand, T., Sullivan, G. J., Bjøntegaard, G., & Luthra, A. (2003). Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7), 560–576. <https://doi.org/10.1109/TCSVT.2003.815165>
- Yamamoto, H. (2000). A new recursive universal code of the positive integers. *IEEE Transactions on Information Theory*, 46(2), 717–723. <https://doi.org/10.1109/18.825820>
- Ziv, J., & Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3), 337–343. <https://doi.org/10.1109/TIT.1977.1055714>
- Ziv, J., & Lempel, A. (1978). Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5), 530–536. <https://doi.org/10.1109/TIT.1978.1055934>