KETIK : Jurnal Informatika
ISSN: 3026-183X (Media Online)
Vol. 01, No. 04, Maret 2024 Hal. 34-47
DOI: 10.70404/ketik.v1i04.61
Faatuatua Media Karya
https://jurnal.faatuatua.com/index.php/KETIK

Perancangan Aplikasi Kompresi Gambar Ekstensi Tiff Dengan Menerapkan Spith Encoding

Liristi Handayani Gea

Universitas Budi Darma, Indonesia, email: liristig@gmail.com

Info Artikel

Diajukan: *26 Maret 2024*
Diterima: *27 Maret 2024*
Diterbitkan: *30 Maret 2024*

Kata Kunci:
Aplikasi;
Kompresi;
Gambar;
Ekstensi;
TIFF;
Spith Encoding.

Keywords:
Applications;
Compression;
Image;
Extensions;
TIFF;
Spith Encoding.

Abstrak

*TIFF adalah salah satu format gambar standar yang digunakan dalam dunia teknologi dan percetakan. Tiff mampu menyimpan kualitas gambar yang sangat tinggi hingga 32 bit .karena file ini mendukung kuallitas yang tinggi ,maka kapasitas file yang dihasilkanjuga sangat besar .hal ini menjadi masalah karena dianggap tidak mendukung untuk keperluan sharing maupun preview.preview sehingga diperlukan suatu Teknik yang dapat mempeerkecil ukuran file Gambar dari Besar Menjadi Kecil Tanpa Mengurangi Kuallitas Video Gambar tesebut Yaitu Dengan Melakukan Image Compression Atau Disebut Juga Kompresi Citra.Pada Penelitian Ini Menggunakan Teknik kompresi citra Terhadap Berkas Gambar Dengan Ekstensi TIFF yang Memiliki Ukuran Gambar yang Sangat Besar. Kompresi data Memiliki Beberapa Algoritma yang Dapat di Hasilkan atau Digunakan Sebuah Aplikasi Kompresi dan Dekompresi dengan Menerapkan SPITH Encoding yang Mampu Memampat Gambar dengan Ekstensi *TIFF dan Menghasilkan Nilai Ratio of Compression (Rc), Compression Ratio Cr), Redudancy (Rd), dan Space Saving (Ss). Kelebihan dari Format * TIFF Mampu Menyimpan Kualitas Gambar yang Sangat Tinggi Hingga 32 bit.*

Abstract

*TIFF is one of the standard image formats used in the world of technology and printing. Tiff is capable of storing very high image quality up to 32 bits. Because this file supports high quality, the resulting file capacity is also very large. This is a problem because it is considered unsupportive for sharing and preview purposes. preview so that a technique is needed that can reduce the size of the image file from large to small without reducing the quality of the video image is by doing Image Compression or also called Image Compression. In this study using image compression techniques against image files with TIFF extensions that have very large image sizes. Data compression has several algorithms that can be generated or used by a compression and decompression application by applying SPITH encoding that is able to compress images with *TIFF extensions and produce ratio of compression (Rc), compression ratio Cr), redudancy (Rd), and space saving (Ss) values. Advantages of *TIFF Format Capable of Storing Very High Image Quality Up to 32 bits.*

.

1. PENDAHULUAN

TIFF adalah salah satu format gambar standar yang digunakan dalam dunia teknologi dan percetakan. Format *.TIFF merupakan singkatan dari kalimat Temporary Instruction File Format atau sering juga disebut Tagged Image File Format. Kelebihan dari format *.TIFF mampu menyimpan kualitas gambar yang sangat tinggi hingga 32 bit. Karena file ini mendukung kualitas yang tinggi, maka kapasitas file yang dihasilkan juga sangat besar. Hal ini menjadi masalah karena dianggap tidak mendukung untuk keperluan sharing maupun preview. Sehingga diperlukan suatu teknik yang dapat memperkecil ukuran file gambar dari besar menjadi kecil tanpa mengurangi kualitas video gambar tersebut yaitu dengan melakukan image compression atau yang disebut juga kompresi citra.

Kompresi citra adalah proses untuk meminimalisasi jumlah bit yang merepresentasikan suatu citra sehingga ukuran data citra menjadi lebih kecil sehingga dapat disimpan atau ditransmisikan secara efisien. Pada penelitian ini penulis menggunakan teknik kompresi citra terhadap berkas gambar dengan ekstensi TIFF yang memiliki ukuran gambar yang sangat besar. Pemilihan teknik kompres citra sebagai solusi untuk mengurangi kapasitas berkas gambar, penulis mengacu pada penelitian terdahulu yang

dilakukan oleh Milpadi Yanti menyatakan bahwa dengan melakukan pemampatan terhadap file gambar maka membuat proses sharing dan preview terhadap gambar menjadi lebih cepat [1]. Namun dalam penerapannya perlu dipilih metode kompresi yang sesuai dengan objek yang di kompresi.

SPITH Encoding merupakan metode kompresi citra yang digunakan oleh penulis pada penelitian ini. Berdasarkan penelitian terdahulu yang dilakukan oleh Nia Maulidia menyatakan bahwa penggunaan SPITH Encoding memiliki nilai kelebihan dari sisi rasio kompresi yang sangat tinggi dan mengkodekan koefisien hasil transformasi gambar dengan memperhatikan kualitas [2]. Selanjutnya penelitian yang dilakukan oleh Amir Said terkait dengan penggunaan SPITH Encoding dalam artikel ilmiahnya menyatakan Implementasi kompresi citra berdasarkan set partitioning in hierarchical trees (SPIHT) memberikan kinerja yang lebih baik dari pada ekstensi metode kompresi citra lainnya. Karena metode ini memiliki prosedur pengkodean dan penguraian baru sangat cepat, dan dapat dibuat lebih cepat lagi, dengan hanya kehilangan kinerja yang kecil, dengan menghilangkan pengkodean entropi dari aliran bit dengan kode aritmatika. Algoritma SPIHT ini menggunakan prinsip pengurutan parsial berdasarkan magnitudo dengan urutan ambang batas yang menurun secara tajam, dan memiliki transmisi bit yang berurutan [3].

Penelitian yang dilakukan penulis yaitu menghasilkan sebuah aplikasi kompresi dan dekompresi dengan menerapkan metode Spith Encoding yang mampu memampat gambar dengan ekstensi *.TIFF dan menghasilkan nilai ratio of compression (Rc), compression ratio (Cr), redundancy (Rd) dan space saving (Ss).

Berdasarkan latar belakang masalah di atas, penulis bermaksud untuk menguraikan tahap-tahap pemberian solusi yang dijelaskan dalam penelitian ini dengan judul “Perancangan Aplikasi Kompresi Gambar Ekstensi TIFF dengan Menerapkan Spith Encoding.”

2. METODE PENELITIAN

2.1 Kerangka Kerja Penelitian

Dalam metodologi penelitian dijabarkan kerangka kerja penelitian atau sering disebut dengan tahapan-tahapan dalam penelitian. Berikut merupakan kerangka kerja dalam proses penelitian dapat dilihat pada gambar 1.

Gambar 1. Kerangka Kerja Penelitian

Kerangka kerja penelitian terdiri dari beberapa tahapan yang saling terkait secara sistematis. Kerangka kerja ini diperlukan untuk mempermudah dalam melakukan suatu penelitian. Berdasarkan gambar kerangka kerja penelitian diatas, maka dapat diuraikan pembahasan dari masing-masing tahapannya sebagai berikut:

1. Tahap identifikasi masalah merupakan tahapan mengidentifikasi masalah untuk mengetahui masalah apa yang dihadapi dalam penyimpanan *file* *.TIFF, sehingga setelah diidentifikasi maka akan dicari solusi untuk memecahkan permasalahan tersebut.
2. Tahap studi pustaka merupakan tahapan untuk mencari data dan informasi yang berkaitan dengan permasalah yang sedang diteliti. Studi pustaka diambil dari berbagai sumber seperti artikel ilmiah, buku dan jurnal yang berkaitan dengan permasalahan yang ingin diselesaikan pada penelitian ini.
3. Tahapan ini dilakukan analisa untuk mengetahui proses kompresi dan dekompresi pada *file* *.TIFF menggunakan algoritma *SIPHT Encoding* serta melakukan penerapan *SIPHT Encoding* yang bertujuan untuk memperkecil ukuran dari *file* *.TIFF itu sendiri.
4. Analisa hasil kompresi merupakan tahapan ini dilakukan presentasi hasil kompresi *file* *.TIFF berdasarkan hasil kompresi dari algoritma *SIPHT Encoding* yang diukur menggunakan parameter menghasilkan nilai *ratio of compression* (Rc), *compression ratio* (Cr), *redundancy* (Rd) dan *space saving* (Ss).
5. Tahap perancangan sistem merupakan tahapan yang memberikan gambaran mengenai perancangan sistem yang akan digunakan untuk mengimplementasikan hasil dari analisa penerapan metode yang telah digunakan sebelumnya. Perancangan sistem yang dirancang adalah membuat aplikasi kompresi *file* *.TIFF dengan menerapkan *SIPHT Encoding* menggunakan GUI Matlab 2019.
6. Tahap implementasi merupakan tahapan ini dilakukan untuk proses pembangunan perangkat lunak yang bertujuan untuk mengetahui apakah sistem dapat berjalan dengan baik dan sesuai dengan kebutuhan atau masih diperlukannya perbaikan pada sistem tersebut.

7. Tahap pengujian merupakan tahapan ini dilakukan pengujian hasil analisa perbandingan algoritma *SIPHT Encoding* pada kompresi *file* *.TIFF berformat *.TIFF proses ini bertujuan untuk mengetahui hasil akhir dari penelitian yang telah dilakukan.

2.2 Kompresi Data

Kompresi data merupakan proses mengurangi jumlah bit yang mempresentasikan suatu data sehingga menghasilkan ukuran file yang lebih kecil [4]. Kompresi bertujuan untuk menghemat penggunaan ruang penyimpanan pada komputer dan juga untuk mempercepat dari pada proses pertukaran data. Kompresi sangat penting karna dapat memperkecil ukuran bytes data, menghemat ruang penyimpanan, dan mempercepat waktu transmisi data. Kompresi memiliki berbagai macam algoritma dan digunakan pada beberapa jenis data seperti file teks, gambar, video, audio, teks dan lain lain. Hasil kompresi dari pada setiap algoritma tidaklah sama tetapi pada dasarnya prinsipnya tetap sama yaitu memperkecil ukuran suatu data.

Dalam melakukan proses kompresi ada beberapa manfaat dari kompresi yaitu sebagai berikut [5]:

1. Pengiriman data lebih cepat
   Proses pengiriman sebuah data menjadi lebih cepat seperti jika ingin mengirimkan gambar, audio, video, dokumen, dan lain lain menjadi lebih cepat karna ukuran dari datanya telah berkurang sehingga proses pengiriman lebih cepat.
2. Media penyimpanan menjadi lebih luas
   Pada saat melakukan kompresi, kapasitas dari media penyimpanan akan menjadi lebih luas sehingga membuat kinerja pada perangkat seperti komputer akan menjadi lebih ringan dan cepat prosesnya.

2.3 Konsep Kompresi Data

Dalam komputer satu karakter dalam code *ASCII (American Standart Code For Information Intercange)* direperentasikan menjadi 8 bit bilangan biner. Jika ternyata jumlah bit data tersebut bukan kelipatan delapan, maka akan dibentuk variabel baru sebagai penambahan ke bit data tersebut agar bit data yang dihasilkan habis dibagi delapan, variabel tersebut adalah padding dan flagging [7]

1. *Padding* adalah penambahan bit 0 sebanyak kekurangan dari jumlah bit data yang terdapat pada hasil proses kompresi, sehingga jumlah keseluruhan bit data tersebut merupakan kelipatan yang habis dibagi delapan (kelipatan delapan). Contoh misalnya bit data yang dihasilkan dari proses kompresi yaitu 100110110011000110100110 1 berjumlah 25 bit data dalam bilangan biner, maka dilakukan penambahan bit 0 sebanyak 7 kali agar jumlah bit data tersebut habis dibagi delapan. Sehingga bit data 10011011001100011010011010000000 menjadi berjumlah 32 bit setelah diberikan *padding.*
2. *Flagging* adalah penambahan bilangan biner sepanjang delapan bit setelah padding. Penambahan flagging ini di maksudkan untuk mempermudah dalam membaca bit data hasil kompresi pada saat proses dekompresi. Misalkan bit data yang telah diberikan padding 10011011001100011010011010000000. Karena terdapat 7 bit penambahan padding maka flagging adalah bilangan biner dari 7 dengan panjang 8 bit yaitu 00000111. Sehingga nilai bit data menjadi 10011011000110100110100000000000111 setelah diberikan *flagging.*

2.4 Pengukuran Kinerja Kompresi Data

Berikut ini adalah ada beberapa kriteria yang biasa digunakan dalam mengukur kualitas sebuah data yang telah dikompresi adalah [3]:

1. *Ration of compression (RC)* merupakan nilai perbamdingan antara ukuran *bit* data sebelum dikompresi dengan ukuran *bit* data yang telah dikompresi.
   $$Rc = \frac{ukuran\ bit\ data\ sebelum\ dikompresi}{ukuran\ bit\ data\ setelah\ dikompresi} \quad \text{.....(1)}$$
2. *Compression Ration (CR)* merupakan presentasi perbandingan antara data yang sudah dikompresi dengan data yang belum dikompresi.
   $$C_R = \frac{ukuran\ bit\ data\ sebelum\ dikompresi}{ukuran\ bit\ data\ setelah\ dikompresi}\ x\ 100\% \quad \text{.....(2)}$$
3. *Space saving (Ss)* merupakan selisih antara data yang belum dikompresi dengan besar data yang sudah dikompresi.
   $$S_S = 100\% - C_R \quad \text{.....(3)}$$

2.5 File Gambar TIFF

TIFF adalah kependekan dari Tagged Image File Format, yang menggunakan kompresi algoritma lossless untuk menyimpan data gambar. Adalah Aldus Corporation yang telah meningkatkan kesadaran TIFF pada tahun 1986. Setelah itu, semakin banyak orang mengabdikan diri untuk mempromosikan versi TIFF. Ekstensi TIFF termasuk TIFF/EP, TIFF/IT, TIFF-F, dan TIFF-FX. Nama lengkap TIFF adalah tagged image file format, yaitu format file komputer untuk menyimpan gambar grafik raster. Orang-orang

telah menerapkan format file gambar yang ditandai ke banyak bidang karena ini adalah format gambar yang banyak digunakan. Industri penerbitan dan foto digital adalah dua tahap utama [3].

Organisasi file TIFF berisi IFH (header file gambar), IFD (direktori file gambar), dan data bitmap dalam tiga bagian. Anda mungkin pernah mendengar BigTIFF sebelumnya, meskipun namanya lucu. BigTIFF adalah salah satu upaya berkelanjutan untuk file TIFF. Tag TIFF adalah deskripsi foto singkat, seperti kode TIFF, nama, nama LibTiff, dan tipe data. Menggunakan format TIF untuk menyimpan foto dapat mempertahankan kualitas gambar yang tinggi dengan pengaruh noise yang rendah. Selain itu, gambar disimpan sebagai format file gambar tag berekstensi TIFF atau TIF. Oleh karena itu, TIFF dan TIF keduanya dapat mewakili file format file yang ditandai [4].

Format TIFF dapat melestarikan gambar fotografi dalam resolusi tinggi. Anda akan menemukan bahwa gambar TIFF akan seperti yang ditunjukkan situs web sumber, yang tidak memiliki penurunan kualitas gambar. Akibatnya, ukuran gambar lebih besar dari JPEG atau format gambar lainnya. Selain itu, file TIFF memiliki kedalaman saluran 16-bit dan 8-bit, dan banyak lapisan dapat disimpan dalam format TIFF tunggal. Namun, Anda perlu menginstal a Penampil TIFF terlebih dahulu sebelum membuka TIFF. TIFF mempertahankan kualitas gambar yang tinggi untuk pencetakan gambar. Begitu banyak fotografer akan memilih TIFF dibandingkan dengan JPEG populer untuk mengedit foto. Oleh karena itu, dalam pencetakan TIFF VS JPEG, TIFF relatif lebih baik daripada JPEG. Profesi selalu menggunakan TIFF untuk mempublikasikan atau merekam objek secara detail.

2.6. Algoritma SPIHT Encoding

Algoritma *SPIHT Endcoding* memiliki empat jenis *codeword* yaitu C1, C2, C3, C4, dan *Codeword* yang digunakan pada penelitian ini adalah C1. Prosedur yang dilakukan untuk mencari *Codeword p*ada algoritma *SPIHT Endcoding* awalnya dimulai dengan mengetahui terlebih dahulu nilai *Unary Code*, B(n) dan $\overline{B}$(n). Untuk mendapatkan nilai *Unary Code* maka perlu diketahui terlebih dahulu bahwa *Unary Code* merupakan bilangan positif dari *n* kemudian *n* -1 dan diikuti oleh satu angka nol. Berikut ini merupakan tebel dari *Unary Code* dapat dilihat pada tabel 1.[3].

Tabel 1. *Unary Code*

n	Code	Code Alternatif
1	0	1
2	10	01
3	110	001
4	1110	0001
5	11110	00001
6	111110	000001
7	1111110	0000001
8	11111110	00000001
9	111111110	000000001
10	1111111110	0000000001

Setelah menemukan nilai *Unary Code* maka selanjutnya mencari B(n) yang merupakan biner dari setiap bilangan bulat *n,* sedangkan $\overline{B}$(n) adalah nilai dari B(n) dengan tanpa adanya bit, artinya adalah setiap angka 1 pada posisi depan dari biner B(n) dihilangkan. Proses selanjutnya setelah menemukan nilai dari $\overline{B}$(n) dan B(n) kemudian barulah dilakukan pencarian nilai *Codeword* C1. Misalnya dalam contoh kasus *n* = 4 dimana nilai biner dari 4 adalah 100 dan jumlah bitnya adalah 3 maka mulai dari kode *unary* 110 dengan menambahkan $\overline{B}$(4) = 00. Hasilnya didapatkan *codeword* lengkapnya adalah 110|00. Berikut ini merupakan *codeword* C1 algoritma *SPIHT Endcoding* dapat dilihat pada tabel 2.

Tabel 2. Codeword SPIHT Endcoding

N	Unary	B(n)	$\overline{B}$(n)	C1
1	0	1		0|
2	10	10	0	10|0
3	110	11	1	10|1
4	1110	100	00	110|00
5	11110	101	01	110|01
6	111110	110	10	110|10
7	1111110	111	11	110|11
8	11111110	1000	000	1110|000
9	111111110	1001	001	1110|001
10	1111111110	1010	010	1110|010

11	11111111110	1011	011	1110|011
12	111111111110	1100	100	1110|100
13	1111111111110	1101	101	1110|101
14	11111111111110	1110	110	1110|110
15	111111111111110	1111	111	1110|111

Langkah-langkah kompresi algoritma *SPIHT Endcoding* adalah sebagai berikut :

1. Menyiapkan *file* *.TIFF yang akan dikompresi
2. Buka *software HxD*, kemudian *input file* *.TIFF yang dikompresi untuk mendapatkan nilai *hexadecimal*
3. Mengambil 20 nilai *hexadecimal* sebagai sampel data
4. Urukan nilai bilangan *hexadecimal* berdasarkan frekuensi kemunculan terbanyak, frekuensi kemunculan paling banyak akan berada diurutan paling atas
5. Pembentukan *Codeword* C1 pada algoritma *SPIHT Endcoding*
6. Kompresi nilai *hexadecimal file* *.TIFF dengan *Codeword* C1 pada algoritma *SPIHT Endcoding*
7. Penyusunan *String Bit*
8. Pemeriksaan *String Bit*
9. Penambahan *Padding* dan *flagging*
10. Pengelompokan *String Bit*
11. Hasil Kompresi Algoritma *SPIHT Endcoding*

Langkah-langkah Dekompresi algoritma *SPIHT Endcoding* adalah sebagai berikut :

1. Mengubah karakter yang dihasilkan dari proses kompresi menjadi nilai biner dan disimal untuk menghasilkan *string bit* semula
2. Pengurangan *string bit*
3. Pembacaan string bit
4. Hasil dekompresi algoritma *SPIHT Endcoding*

3. HASIL DAN ANALISIS

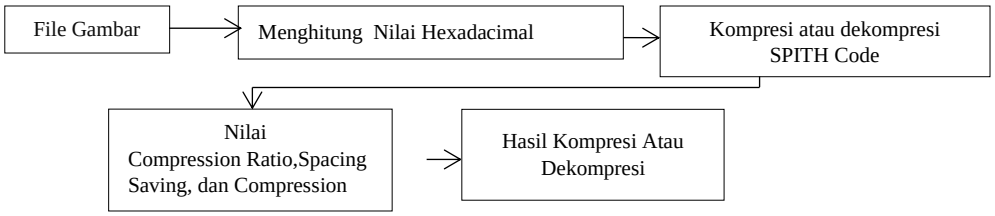
3.1 Analisa Penerapan Metode

Penerapan metode pada penelitian ini mengacu pada judul penelitian Perancangan Aplikasi Kompresi Gambar Ekstensi TIFF dengan Menerapkan *Spith Encoding*.

Berikut tahapan penerapan metode SPITH Encoding dalam proses kompresi atau dekompresi gambar dengan ekstensi *.TIFF:

1. Menetapkan file gambar dengan ekstensi *.TIFF yang di kompresi atau di dekompresi.
2. Melakukan ekstrasi nilai *hexadecimal* pada gambar dengan mengunakan aplikasi *Binary Viewer*.
3. Melakukan proses kompresi atau dekompresi sesuai dengan tahapan penerapan *SPITH Encoding*.
4. Selanjutnya menganalisis hasil kompresi atau dekompresi dengan melihat nilai *Compression Rasio*, *Spacing Saving* dan *Compression of Rasio*

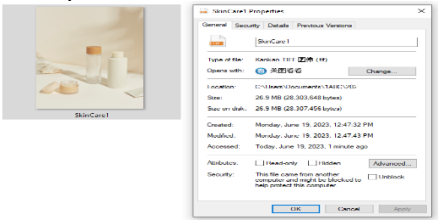
Berdasarkan prosedur kompresi dan dekompresi file gambar *.TIFF di atas, lebih jelasnya dapat dilihat dalam bentuk gambar diagram di bawah ini.



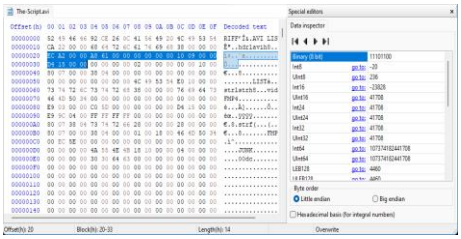
Gambar 2. Prosudur Kompresi dan Dekompresi File Mp4

3.2 Penerapan SPITH Code

Sebelum *file* *.*.TIFF dikompresi, terlebih dahulu dilakukan pembacaan biner untuk mendapatkan nilai *hexadecimal* dengan menggunakan aplikasi *hxd editor*.



Gambar 3. Sampel Data



Gambar 4. Nilai Hexadecimal

Algoritma *SPITH Code* memiliki teknik yang baik terhadap kompresi *file* seperti *file* teks, video, audio maupun gambar. Penulis menerapkan algoritma ini ke dalam penelitian kompresi, berikut ialah proses kerja dari algoritma *SPITH Code* di bawah ini.

1. Analisa Proses Kompresi

   Sebelum melakukan kompresi, terlebih dahulu perlu pembacaan nilai *hexadecimal file* gambar. Berikut nilai *hexadecimal* dari *sample file* gambar, dapat diketahui pada tabel di bawah ini.

Tabel 3. Tampilan Nilai Hexa

EC	A2	00	00	A8	61	00	00	00	00
00	00	10	09	00	00	D4	15	00	00

Sesuaitabel di atas, didapat nilai bilangan *hexadecimal file* gambar berekstensi **.AVI* yang dihitung secara manual dengan mengambil 20 sampel nilai *hexadecimal.* Berikut nilai bilangan *hexadecimal* diurutkan sesuai frekuensinya, nilai frekuensi paling banyak yang akan berada diurutan pertama.

Tabel 4. Nilai Hex Yang Belum Dikompresi

N	Hexadecimal	Biner	Frekuensi	Bit	Frekuensi x Bit
1	00	00000000	12	8	96
2	EC	11101100	1	8	8
3	A2	10100010	1	8	8
4	A8	10101000	1	8	8
5	61	01100001	1	8	8
6	10	00010000	1	8	8
7	09	00001001	1	8	8
8	D4	11010100	1	8	8
9	15	00010101	1	8	8
		Total Bit			160 Bit

Setelah bilangan *hexadecimal* diurutkan sesuai dengan frekuensi kemunculan serta didapat nilai biner, maka selanjutnya menghitung *bit* dan melakukan pengurutan kode algoritma *SPITH Code* dan memperoleh *bit file* terkompresi. Adapun proses kompresi *file* gambar dapat dilihat pada tabel di bawah ini:

Tabel 5. Pengurutan Kode SPITH Code

N	*Hexa*	*Codeword*	Bit	Freq	Bit x Freq
1	00	1010	4	12	48
2	EC	0101	4	1	4
3	A2	1011	4	1	4
4	A8	010010	6	1	6
5	61	100100	6	1	6
6	10	010011	6	1	6
7	09	100101	6	1	6
8	D4	100111	6	1	6
9	15	100110	6	1	6
		Total Bit			92

Setelah kode berhasil diurutkan sesuai dengan perhitungan kode *SPITH Code,* kemudian langkah selanjutnya ialah menyusun kembali *string bit* yang telah dihasilkan dari proses kompresi sesuai dengan karakter pada nilai *hexadecimal.*

Tabel 6. String Bit Hasil Kompresi

EC	**A2**	**00**	**00**	**A8**

0101	1011	1010	1010	010010
61	**00**	**00**	**00**	**00**
100100	1010	1010	1010	1010
00	**00**	**10**	**09**	**00**
1010	1010	010011	100101	1010
00	**D4**	**15**	**00**	**00**
1010	100111	100110	1010	1010

Dari tabel di atas, *string bit* hasil kompresi *file* gambar menggunakan algoritma *SPITH Code* dapat dituliskan sebagai berikut: "0101101110101010010010100100101010101010101010101010010011100101101010101001111 0011010101010" Sebelum didapatkan hasil akhir keseluruhan kompresi akan dilakukan penambahan *string bit* yaitu *padding* dan *flagging* dengan mengacu pada sisa jumlah *bit* dibagi 8. Karena dari jumlah bit x frekuensi yang bernilai 92 tidak habis dibagi 8 maka akan dibentuk variabel untuk penambahan *bit* data. *Padding* adalah penambahan *bit* data agar seluruh jumlah *bit* data tersebut kelipatan 8. Rumus *padding* yaitu $7 - n + "1"$. Sedangkan *flagging* adalah penambahan bilangan *biner* setelah *padding* untuk mempermudah membaca bit-bit hasil kompresi. Rumus *flagging* yaitu $9 - n$.

Tabel 7. Penambahan *Padding* dan *Flagging*

Padding	Flagging
92 *mod* 8 = 4 = *n* 7 – n + "1" 7 – 4 + "1" = 0001	9 – n 9 – 4 = 5 = 00000101

Jadi *string bit* yang dihasilkan setelah ditambah *padding* dan *flagging* yakni: "0101101110101010010010100100101010101010101010101010010011100101101010101001111 0011010101010000100000101" Sehingga total panjang *bit* setelah ditambahkan *padding* dan *flagging* adalah "104". Selanjutnya lakukan pemisahan *bit* menjadi beberapa kelompok. Setiap kelompok terdiri dari 8 *bit*.

01011011	10101010	01001010	01001010	10101010
10101010	10100100	11100101	10101010	10011110
01101010	10100001	00000101		

Berdasarkan pada hasil pengelompokkan *bit,* maka terdapat 13 kelompok nilai biner baru yang sudah terkompresi beserta penambahan nilai *biner*. Setelah itu, langkah selanjutnya ialah dengan mengubah nilai *biner* ke nilai *hexadecimal* dan *decimal* untuk mengetahui karakter yang sesuai dengan nilai *biner* berdasarkan pada tabel *ASCII*. Berikut nilai *hexadecimal* yang sudah dikompresi dapat diketahui pada tabel di bawah ini.

Tabel 8. Hasil Karakter Terkompresi

No	*Codeword* Hasil Kompresi	*Decimal*	*Hexadecimal*	Karakter
1.	01011011	93	133	[
2.	10101010	170	AA	a
3.	01001010	74	4A	J
4.	01001010	74	4A	J
5.	10101010	170	AA	a
6.	10101010	170	AA	a
7.	10100100	164	A4	¤
8.	11100101	229	E5	å
9	10101010	170	AA	a
10	10011110	158	9E	ž
11	01101010	106	6A	j
12	10100001	161	A1	i
13	00000101	5	05	ENQ

Selanjutnya untuk menentukan parameter kinerja dari algoritma kompresi, maka penulis menentukan hasil kompresi *file* gambar sesuai dengan parameter kinerja kompresi yang ditetapkan.

Ukuran data sebelum dikompresi yaitu 160/8 = 20 byte
Ukuran data sesudah dikompresi yaitu 104/8 = 13 byte
Sesuai data di atas dapat dihitung parameter kinerja kompresinya seperti di bawah ini:

a. *Ratio Of Compression* (Rc)

= $\frac{160}{104}$ = 1,5384

b. *Compression Rasio* (Cr)
= $\frac{104}{160}$ * 100 % = 65%

c. *Space Savings* (Ss)
= 100% - 65%
= 35%

d. *Redudancy*
= $\frac{160-104}{160} \times 100\%$
= 35%

Berdasarkan presentasi rasio kompresi diatas maka kapasitas file gambar awal 26,9 mb berkurang menjadi :
= 26,9 mb x 65 %
= 17,5 mb
Berdasarkan nilai tersebut maka kapasitas yang terkompresi sebanyak 17,5 mb dari 26,9 mb, sehinggs file gambar akhir berkapasitas9,4mb.

2. Analisa Proses Dekompresi
Proses dekompresi akan dilakukan pada *file* gambar yang telah dikompresi sebelumnya dan dapat dilakukan dengan mengembalikan karakter menjadi *hexadecimal* kemudian diubah menjadi *string bit* pemula.

Tabel 9. Nilai Biner dan Decimal

No	Karakter	Biner	Decimal
1.	[	01011011	93
2.	A	10101010	170
3.	J	01001010	74
4.	J	01001010	74
5.	a	10101010	170
6.	a	10101010	170
7.	¤	10100100	164
8.	å	11100101	229
9	a	10101010	170
10	ž	10011110	158
11	j	01101010	106
12	i	10100001	161
13	ENQ	00000101	5

Sesuai dengan tabel nilai *decima*l dan karakter di atas, selanjutnya diambilkan nilai keseluruhan *biner* kemudian digabungkan menjadi:
“01011011101010100100101001001010101010101010101010100100111001011010101010011110011010101010000100000101”
Pada saat pengurangan *string bit* menjadi *string bit* semula dengan menghilangkan biner *padding* dan *flagging.* Untuk mengembalikan *string bit* menjadi *string bit* sebelumnya dapat dilakukan dengan pembacaan 8 *bit* terakhir, pembacaan berupa bilangan *decimal.* Nyatakan hasil pembacaan dengan n , gunakan rumus $7 + n$ untuk mengembalikan *string bit* ke bentuk semula.
8 *bit* terakhir = 00000101 = 5 = n
7 + n = 7 + 5 = 12
“01011011101010100100101001001010101010101010101010100100111001011010101010011110011010101010000100000101” hilangkan dari *string bit* sebanyak 12 *bit* terakhir, sehingga menjadi:
“01011011101010100100101001001010101010101010101010100100111001011010101010011110011010101010”
Sesuai perhitungan yang sudah dilakukan, maka mendapatkan hasil *string bit* berjumlah 92 *bit* seperti diawal, kemudian melakukan pembacaan *string bit* awal. Pembacaan *string bit* dengan algoritma *SPITH Code* dilakukan dengan memakai *Bruto Force.* Dimana *string bit* dibaca dengan *index* terendah yang kemudian dilanjutkan ke *index* setelahnya yang tidak mewakili karakter algoritma *SPITH Code.* Berikut tabel dari pengecekan *bit* yang sudah dikompresi sebelumnya.

Tabel 10. Pemeriksaan Bit Dekompresi

No	Nilai Biner	Keterangan	Nilai Hexadecimal
1.	0	Tidak Ada	
2.	01	Tidak Ada	
3.	010	Tidak Ada	
4.	0101	Ada	EC
5.	1	Tidak Ada	
6.	10	Tidak Ada	
7.	101	Tidak Ada	
8.	1010	Ada	A2
9.	1	Tidak Ada	
10.	10	Tidak Ada	
11.	101	TidakAda	
12.	1010	Ada	00
13.	1	Ada	
14.	10	Tidak Ada	
15.	101	Tidak Ada	
16.	1010	Tidak Ada	00
17.	0	Tidak Ada	
18.	01	Ada	
19.	010	Tidak Ada	
20.	0100	Tidak Ada	
21.	01001	Tidak Ada	
22.	010010	Ada	A8
23.	1	Tidak Ada	
24..	10	Tidak Ada	
25.	100	Tidak Ada	
26.	1001	Tidak Ada	
27.	10010	Tidak Ada	
28.	100100	Ada	61
29.	1	Tidak Ada	
30.	10	Tidak Ada	
31.	101	Tidak Ada	
32.	1010	Tidak Ada	00
33.	1	Ada	
34.	10	Tidak Ada	
35.	101	Tidak Ada	
36.	1010	Tidak Ada	00
37.	1	Tidak Ada	
38.	10	Tidak Ada	
39.	101	Tidak Ada	
40.	1010	Ada	00
41.	1	Tidak Ada	
42.	10	Tidak Ada	
43.	101	Tidak Ada	
44.	1010	Ada	00
45.	1	Tidak Ada	
46.	10	Tidak Ada	
47.	101	Tidak Ada	
48.	1010	Ada	00
49.	1	Tidak Ada	
50.	10	Tidak Ada	
51.	101	Tidak Ada	
52.	1010	Ada	00

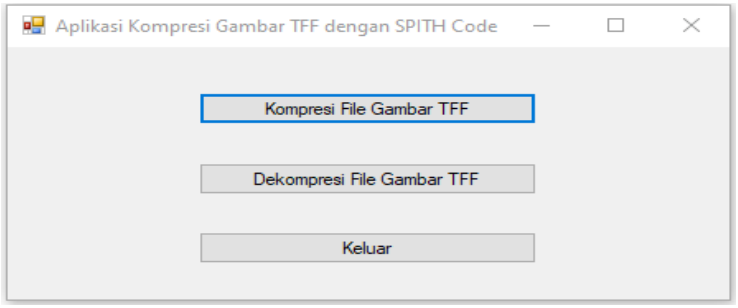
No	Nilai Biner	Keterangan	Nilai Hexadecimal
53.	0	Tidak Ada	
54.	01	Tidak Ada	
55.	010	Tidak Ada	
56.	0100	Tidak Ada	
57.	01001	Tidak Ada	
58.	010011	Ada	10
59.	1	Tidak Ada	
60.	10	Tidak Ada	
61.	100	Tidak Ada	
62.	1001	Tidak Ada	
63.	10010	Tidak Ada	
64.	100101	Ada	09
65.	1	Tidak Ada	
66.	10	Tidak Ada	
67.	101	Tidak Ada	
68.	1010	Ada	00
69.	1	Tidak Ada	
70.	10	Tidak Ada	
71.	101	Tidak Ada	
72.	1010	Ada	00
73.	1	Tidak Ada	
74.	10	Tidak Ada	
75.	100	Tidak Ada	
76.	1001	Tidak Ada	
77.	10011	Tidak Ada	
78.	100111	Ada	D4
79.	1	Tidak Ada	
80.	10	Tidak Ada	
81.	100	Tidak Ada	
82.	1001	Tidak Ada	
83.	10011	Tidak Ada	
84.	100110	Ada	15
85.	1	Tidak Ada	
86.	10	Tidak Ada	
87.	101	Tidak Ada	
88.	1010	Ada	00
89.	1	Tidak Ada	
90.	10	Tidak Ada	
91.	101	Tidak Ada	
92.	1010	Ada	00

Sesuai dari hasil tabel pemeriksaan dekompresi yang telah dilakukan di atas maka didapatkan nila *hexadecimal* sama seperti hasil kompresi sebelumya yakni: **EC A2 00 00 A8 61 00 00 00 00 00 00 10 09 00 00 D4 15 00 00.** Setelah dekompresi *file* telah dilakukan akan kembali ke *file* asli atau sebelumnya dan tidak menghilang data dari ukuran *file* sebelum dikompresi.

3.3 Tampilan Aplikasi

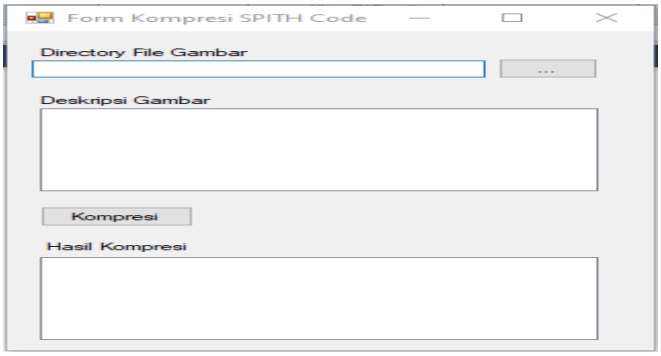
Tampilan Sistem merupakan tampilan akhir dari antar muka sistem yang di rancang, pada sistem ini terdapat 4 halaman yaitu form utama aplikasi, form kompresi dan form dekompres.

1. Form Utama Aplikasi



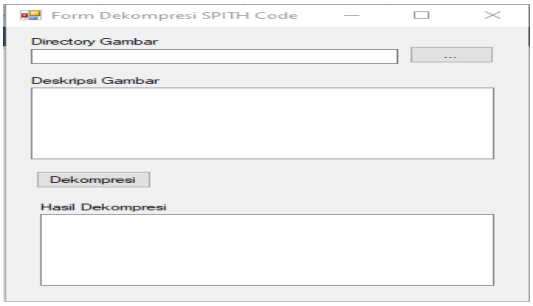
Gambar 5. Tampilan Antarmuka Form Utama Aplikasi

2. Form Kompresi



Gambar 6. Tampilan Antarmuka Form Kompresi

3. Form Dekompresi



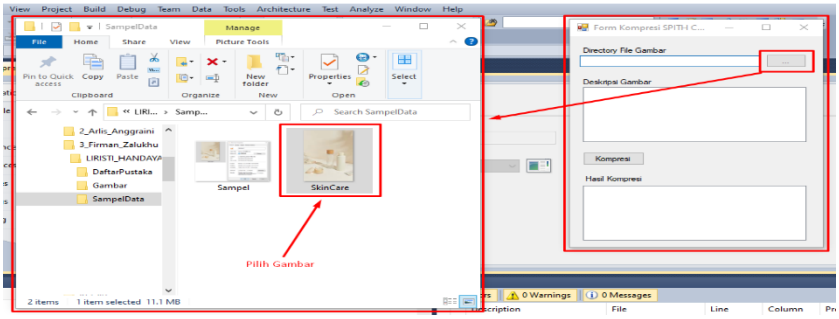
Gambar 7. Tampilan Antarmuka Form Dekompresi

3.4 Pengujian Aplikasi

Tujuan pengujian sistem adalah untuk mengetahui bagaimana sistem yang dibangun bekerja. Sistem ini dirancang sesederhana mungkin agar pengguna dapat mengoperasikannya dengan mudah. Berikut ini tampilan *print out* saat sistem sedang berjalan.

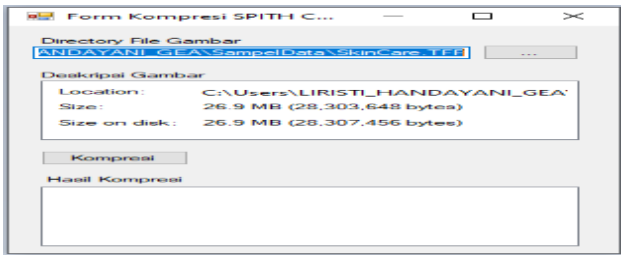
1. Proses Penggunaan Form Kompresi

Form ini menampilkan bagaimana prose kompresi gambar dengan menggunakan SPITH Code, dimana user terlebih dahulu memasukkan data gambar yang akan di kompresi, seperti pada gambar berikut ini:



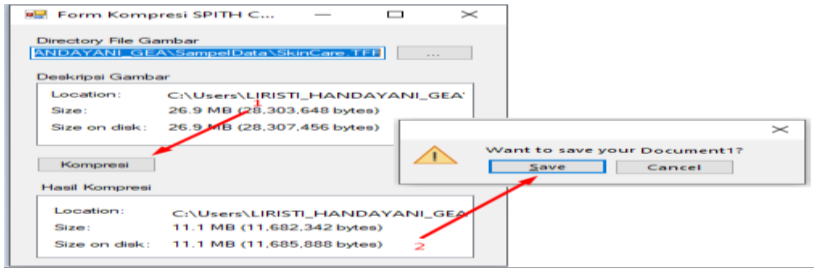
Gambar 8. Tampilan Proses Akses Gambar Kompresi

Selanjutnya pilih gambar dengan ekstensi *.TFF untuk dikompresi, sehingga tampilan form kompresi tampak seperti pada gambar berikut ini:



Gambar 9. Proses Pembacaan Gambar *.TFF

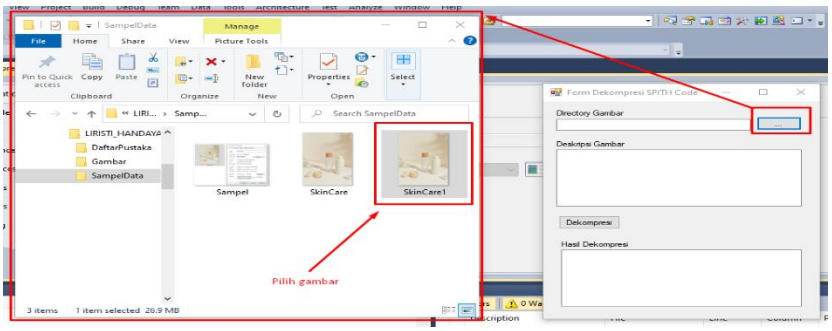
Selanjutnya tekan tombol Kompresi untuk melakukan proses kompresi SPITH Code yang telah ditanamkan dalam tombol "Kompresi" dan secara otomatis muncul proses penyimpanan gambar yang terkompres klik save maka gambar tersimpan dalam *directory*, lebih jelasnya dapat dilihat pada tampilan gambar berikut ini:



Gambar 10. Hasil Kompresi *.TFF

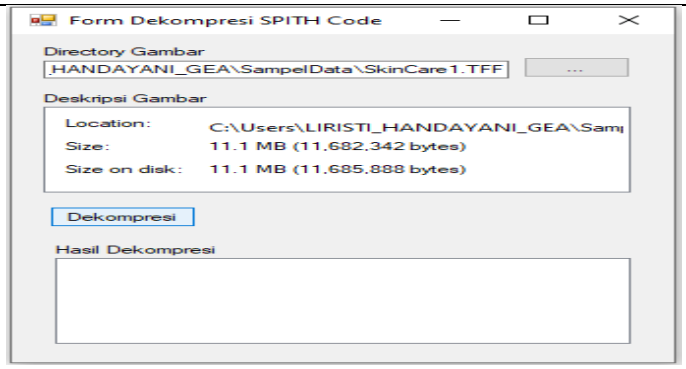
2. Proses Penggunaan Forma Dekompresi

Form ini menampilkan bagaimana prose dekompresi gambar dengan menggunakan SPITH Code, dimana user terlebih dahulu memasukkan data gambar yang akan di dekompresi, seperti pada gambar berikut ini:



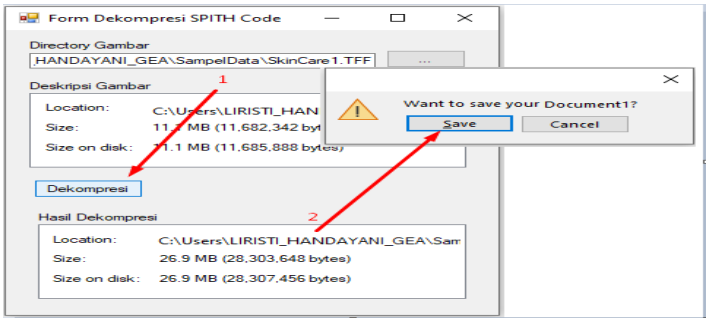
Gambar 11. Tampilan Proses Akses Gambar Dekompresi

Selanjutnya pilih gambar dengan ekstensi *.TFF untuk di dekompresi, sehingga tampilan form dekompresi tampak seperti pada gambar berikut ini:



Gambar 12. Tampilan Proses Akses Gambar Dekompresi

Tahap selanjutnya tekan tombol Dekompresi untuk melakukan proses dekompresi SPITH Code yang telah ditanamkan dalam tombol "Dekompresi" dan secara otomatis muncul proses penyimpanan gambar yang di dekompres, klik save maka gambar tersimpan dalam *directory*, lebih jelasnya dapat dilihat pada tampilan gambar berikut ini:



Gambar 13. Hasil Dekompresi *.TFF

4. KESIMPULAN

Berdasarkan penelitian yang telah dilakukan, dapat ditarik beberapa kesimpulan dari pembahasan sebelumnya dari hasil akhir penelitian ini, kesimpulannya adalah sebagai berikut:

1. Sampel data yang digunakan berukuran 26,9 MB. Dari *file* yang menjadi sampel data, maka diambil sampel data sebesar 20 bilangan *hexadecimal* yang terdiri dari 20 *bytes* yang dimana setiap *bytes* terdiri dari 8 *bit.* Jadi, dapat dihitung jumlah *bit* yang diproses adalah 20 x 8 *bit* atau sebanyak 160 *bit.*
2. Prosedur kompresi citra dengan ekstensi *.TIFF dapat dilakukan dengan cara ilakukan dengan mengkonversi ke format hexadecimal untuk melakukan perhitungan. Angka hexadecimal yang dihasilkan diproses dengan menggunakan SPITH Encoding untuk menghasilkan nilai baru yang mengubah ukuran file citra
3. Penerapan PITH Encoding dalam proses pemampatan gambar ekstensi *. TIFF menghasilkan nilai nilai Ratio of Compression (Rc) sebesar 1,5384, Compression Ratio (Cr) 65%, Redudancy (Rd) 35% dan Space Saving 35%.
4. Aplikasi kompresi file video menggunakan Spith Encoding (P2) yang dirancang menggunakan Microsoft Visual Studio 2008 dapat digunakan dan dijalankan dengan baik dalam mengkompresi file gambar ekstensi *. TIFF.

REFERENSI

[1] M. Yanti, "Aplikasi Kompresi Citra Dengan Menerapkan Algoritma SPIHT (Set Partitioning In Hierarchical Trees)," vol. 1, no. 1, 2021.

[2] N. Maulidia, "Analisa Pengiriman Citra Terkompresi SPIHT dengan Teknik Spread Spectrum Direct Sequence (DS-SS)," vol. 1, no. 1, 2015.

[3] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 243–250, Jun. 1996, doi: 10.1109/76.499834.

[4] A. Pradana, "Analisa Perbandingan Algoritma Elias Gamma Code Dan Algoritma Goldbach Code Pada Kompresi File Dokumen," vol. 6, 2022.

[5] E. S. Sitorus, "Analisis Perbandingan Algoritma Goldbach G0 Dan Algoritma Goldbach G1 Dalam Mengkompresikan File Gambar," vol. 6, 2022.

[6] E. N. Simanjuntak, “Penerapan Algoritma Prefix Code Pada Kompresi File Gambar,” vol. 1, no. 3, 2021.
[7] Y. Anggraini, “Perancangan Aplikasi Kompresi File PDF Dengan Menerapkan Algoritma Golomb,” vol. 6, 2022.
[8] D. Pratiwi and T. Zebua, “Analisis Perbandingan Kinerja Algoritma Fixed Length Binary Encoding Dan Algoritma Elias Gamma Code Dalam Kompresi File Teks,” *KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer)*, vol. 3, no. 1, pp. 424–430, 2019, doi: 10.30865/komik.v3i1.1623.
[9] Mawar, “Perancangan Aplikasi Kompresi File Pdf Dengan Menerapkan Algoritma Punctured Elias Codes Mawar,” *Informasi dan Teknologi Ilmiah (INTI)*, vol. 7, no. 3, pp. 217–223, 2020.
[10] S. R. Saragih and D. P. Utomo, “Penerapan Algoritma Prefix Code Dalam Kompresi Data Teks,” *KOMIK (Konferensi Nasional …*, vol. 4, pp. 249–252, 2020, doi: 10.30865/komik.v4i1.2691.
[11] I. Ihsan and D. P. Utomo, “Analisis Perbandingan Algoritma Even-Rodeh Code Dan Algoritma Subexponential Code Untuk Kompresi File Teks,” *KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer)*, vol. 4, no. 1, pp. 223–227, 2020, doi: 10.30865/komik.v4i1.2684.