SinkrOn
JURNAL & PENELITIAN TEKNIK INFORMATIKA
Jurnal & Penelitian Teknik Informatika
Volume 1 Nomor 1, Oktober 2016
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Optimasi Rasio Kompresi Dan Kompleksitas Waktu Kompresi File Teks Menggunakan Algoritma *Lempel-ZIV-Welch* Dengan *Fibonacci Search*

Yonata Laia[1]
[1]Universitas Prima Indonesia
Medan
[1]yonatan_li@yahoo.co.id

Mardi Turnip[2]
[2]Universitas Prima Indonesia
Medan
[2]mardyturnip@gmail.com

Abstract — Kompresi data berarti suatu teknik untuk memampatkan data agar diperoleh data dengan ukuran yang lebih kecil daripada ukuran aslinya sehingga lebih efisien dalam menyimpan serta mempersingkat waktu pertukaran data tersebut. Algoritma LZW ini dirancang cepat tetapi tidak bisa bekerja optimal karena hanya melakukan analisis terbatas pada data. Penelitian yang berkaitan dengan algoritma LZW menyatakan bahwa rasio kompresi dan waktu kompresi kurang optimal, serta algoritma LZW ini memerlukan waktu yang sangat besar dalam mengkompresi data. Pada penelitian ini, Algoritma *Fibonacci Search* (FS) diterapka untuk meningkatkan kinerja algoritma LZW dalam hal pencarian kamus kata (*Dictionary*) sehingga dengan pencarian yang optimal akan diperoleh kinerja LZW yang lebih baik. Data yang di gunakan dalam penelitian ini adalah data teks dengan alasan karena data teks lebih sederhana dalam pemrosesannya. Dari hasil penelitian dengan menguji lima jenis data teks menggukan algoritma LZW, dengan kapasitas yang berbeda yaitu 6,9,12,19 dan 24 Kb di peroleh rata – rata ukuran file kompresi sebesar 4, 148, rata-rata waktu kompresi sebesar 118,8, rata-rata rasio kompresi 64%. Dengan metode LZWFS di peroleh rata-rata ukuran file kompresi 5,126, rata-rata waktu kompresi 86,2, rata – rata rasio kompresi 58%. Dari hasil penelitian diatas di peroleh kesimpulan bahwa Algoritma LZWFS berhasil menyingkat waktu pencarian data namun masih memiliki kelemahan pada saat pengurutan data.

Keywords — *Kompresi LZW, Rasio,* Kompleksitas Waktu, *Fibonacci Search, File Teks.*

I. PENDAHULUAN

Salah satu kegunaan kompresi adalah untuk memperkecil ukuran file dalam memori atau media penyimpanan, agar media tersebut dapat di hemat. Kompresi data berarti suatu teknik untuk memampatkan data agar diperoleh data dengan ukuran yang lebih kecil daripada ukuran aslinya sehingga lebih efisien dalam menyimpan serta mempersingkat waktu pertukaran data tersebut. Beberapa software kompresi yang banyak digunakan para pengguna komputer saat ini diantaranya adalah WinZip (menghasilkan format.zip) dan WinRAR (menghasilkan format.rar) dan lainnya. [1].

Algoritma LZW *(Lempel-Ziv-Welch)* banyak digunakan untuk mengkompresi data digital [2], seperti file gambar [3], file teks [4]. Ada beberapa penelitian yang sudah membandingkan performan algoritma *LZW* dengan algoritma yang sejenis misalnya *Huffman , RLE, Shannon-Fano* [5] meneliti perbandingan algoritma kompresi terhadap objek citra. Hasil penelitian menunjukkan bahwa Algoritam LZW menampilkan hasil kompresi yang lebih baik di banding dengan algoritma *Huffman, Shannon-Fano Dan REL*. [6] meneliti performan kompresi huffman dan LZW pada WSN (*Wireles Sensor Node*) performan Huffman yang lebih bagus.

Algoritma LZW ini dirancang cepat tetapi tidak bisa bekerja optimal karena hanya melakukan analisis terbatas pada data [7]. Peneliti yang lain [8] menyatakan bahwa rasio kompresi dan waktu kompresi kurang optimal. [9] juga menyatakan kalau algoritma LZW ini memerlukan waktu yang sangat besar dalam mengkompresi data.

Beberapa peneliti menerapkan beberapa metode untuk memperbaiki kinerja LZW. [10] menerapkan metode BPS (*Bit Plane Slicing*). Hasil penelitian menunjukkan bahwa diperoleh kinerja LZW yang baik dalam mengkompres. [11].
Menerapkan BS (*Binary Search*). Hasil penelitian menujukkan bahawa kinerja LZW yang lebih baik.

Fibonacci Search (FS) adalah salah satu algoritma pencarian data yang sudah umum digunakan. Beberapa penelitian terdahulu tentang penerapan metode FS menunjukkan bahwa algoritma FS mampu memperbaiki kinerja sebuah sistem secara hibrida, [12]. menerapkan Fibonacci Search pada pencocokan (*matching*) blok video, [13] menerapkan Fibonacci Search pada teknik pencarian garis (*Line Search*). Hasil penelitian menunjukkan bahwa penerapan *Fibonacci Search* (FS)

dalam pencarian secara non linear menghasilkan kecepatan yang tinggi.

Pada penelitian ini, Algoritma *Fibonacci Search* (FS) diterapka untuk meningkatkan kinerja algoritma LZW dalam hal pencarian kamus kata (*Dictionary*) sehingga dengan pencarian yang optimal akan diperoleh kinerja LZW yang lebih baik.

II. TUJUAN PUSTAKA

A. *Kompresi*

Kompresi terdiri dari dua komponen, algoritma *encoding* yang mengambil pesan dan menghasilkan sebuah representasi "kompresi" (diharapkan dengan sedikit bit), dan algoritma *decoding* yang merekonstruksi pesan asli atau perkiraan dari representasi kompresi. [14].

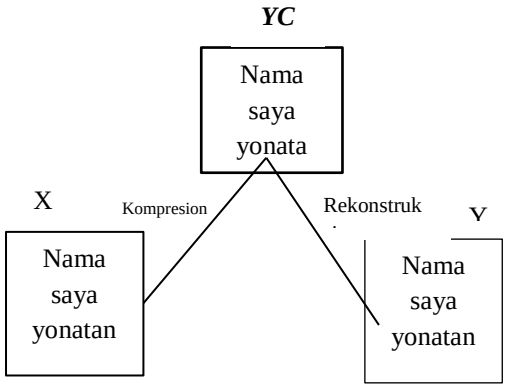
Di dalam penelitian ini, file yang akan diteliti berfokus pada jenis file teks, dengan ekstensi file txt (flat file) adalah salah satu jenis file komputer yang tersusun dalam suatu urutan baris data teks biasanya diwakili oleh 8 bit kode ASCII [15].

B. *Teknik Kompresi*

Ketika kita berbicara tentang teknik kompresi atau algoritma kompresi, kita sebenarnya mengacu pada dua algoritma. Ada algoritma kompresi yang mengambil sebuah input x dan menghasilkan x_c sebagai representasi yang

Berdasarkan persyaratan rekonstruksi, skema kompresi data dapat dibagi menjadi dua kelas besar: Skema kompresi *lossless*, di mana x identik dengan y, dan skema kompresi *lossy*, yang umumnya menyediakan kompresi jauh lebih tinggi daripada kompresi lossless tetapi memungkinkan menjadi berbeda dari x .

memerlukan bit yang lebih sedikit, dan ada algoritma rekonstruksi yang beroperasi pada representasi kompresi x_c untuk menghasilkan rekonstruksi y. Operasi ini secara skematis diperlihatkan pada Gambar 2.1 [17].



Gambar 1. Kompresi dan Rekonstruksi (Sayood, 2006)

C. *Kompresi lossy (lossy compression)*

Lossy compression menyebabkan adanya perubahan data dibandingkan sebelum dilakukan proses kompresi. Sebagai gantinya lossy compression memberikan derajat kompresi lebih tinggi. Tipe ini cocok untuk kompresi file suara digital dan gambar digital. File suara dan gambar secara alamiah masih bisa digunakan walaupun tidak berada pada kondisi yang sama sebelum dilakukan kompresi [16].

D. *Metode lossless*

Kompresi *Lossless* adalah kelas dari algoritma data kompresi yang memungkinkan data yang asli dapat disusun kembali dari data kompresi. Lossless data kompresi digunakan dalam berbagai aplikasi seperti format Rar, ZIP dan GZIP. *Lossless* juga sering digunakan sebagai komponen dalam teknologi kompresi data *lossy*. Kompresi *Lossless* digunakan ketika sesuatu yang penting pada kondisi asli.

III. HASIL DAN PEMBAHASAN

A. *Data yang di gunakan*

Data yang di gunakan dalam penelitian ini dalam data teks atau data yang ekstensi txt dapat dilihat pada tabel I alasan peneliti mengapa memilih data teks yang di gunakan adalah karena data teks untuk memprosesnya sederhana.

Tabel 1. Kapasitas Teks Yang Di Gunakan

NO	*Nama File*	*Ukuran*
1	Test1.txt	2 kb
2	Test2.txt	5 kb
3	Test3.txt	9 kb
4	Test4.txt	10 kb
5	Test5.txt	20 Kb

B. *Proses Algoritma LZW*

Sebagai contoh, String “ABACCACD” akan di kompresi dengan LZW. Isi *dictionary* pada awal di set dengan tiga karekater dasar yang ada : “A”, ”B”, ”C” dan “D”. Tahapan kompresi di jelaskan pada Tabel 3.2

STRING= ABACCACD

Dari string diatas didapat kamus awal dengan menguji setiap string satu persatu dan memasukkan kekamus awal apabila belum ditemukan sebelumnya, dapat dilihat pada tabel dibawah ini:

e-ISSN : 2541-2019
p-ISSN : 2541-044X

Tabel 2. Tahap Pencarian Kamus Awal

INDEX	*KAMUS*
[1]	A
[2]	B
[3]	C
[4]	D

Setelah kamus awal ditemukan lalu gabung 2 karakter dalam string dan uji apakah sudah ada karakter gabungan tersebut dalam kamus awal, apabila gabungan karakter sebelum ditemukan maka masukan sebagai kamus tambahan, dan apabila sudah ada maka tambah satu karakter berikutnya lalu uji kembali apakah sudah ada atau belum pada kamus awal dan kamus tambahan proses kamus tambahan dapat dilihat pada tabel 3

Tabel 3. Tahap Pencarian Kamus Tambahan

Langkah	String 1	String 2	*Dictionary*	Output
1	A	B	[5] A B	1
2	B	A	[6] B A	2
3	A	C	[7] A C	1
4	C	C	[8] C C	3
5	C	A	[9] C A	3
6	A	C	-	1
7	AC	D	[10]ACD	-
8	-	-	-	10

Kolom *String* 1 menyatakan karakter awal yang akan digabung dan string 2 karakter selanjutnya. Hasil dari penggabungan string 1 dan string 2 yang belum terdapat pada dictionary akan dijadikan kamus yang baru dan nilai kamus pada string 1 merupakan output yang di jadikan hasil dari perhitungan LZW. Contoh Hasil Penggabungan string 1 dan 2 pada langkah 1 = "AB" dapat dilihat pada tabel IV.

Tabel 4. Tahap Pencarian Karakter ke-1 Kompresi LZW

Dictionary	UJI	*Output*
[1]A	!=AB	
[2]B	!=AB	
[3]C	!=AB	
[4]D	!=AB	
Kamus Baru [5]AB		1

Dan apabila data hasil penggabungan string 1 dan string 2 sudah ada pada kamus maka hasil penggabungan menjadi string satu pada langkah selanjutnya dan output dianggap tidak ada. Contoh hasil penggabungan string 1 dan 2 pada langkah 6="AC" dapat dilihat pada tabel 5.

Tabel 5. Tahap Pencarian Karakter ke-2 dengan Kompresi LZW

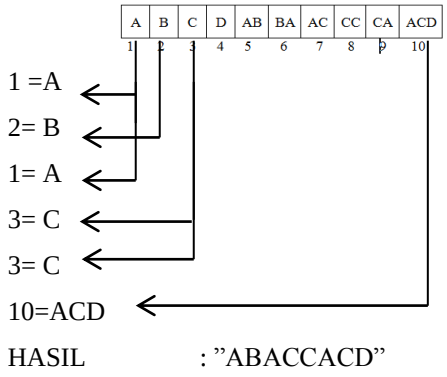
Dictionary	UJI	*Output*
[1]A	!=AC	-
[2]B	!=AC	-
[3]C	!=AC	-
[4]D	!=AC	-
[5] A B	!=AC	1
[6] B A	!=AC	2
[7] A C	=AC Ditemukan	1
[8] C C	-	3
[9] C A	-	3
-	-	-
[10]ACD		10

B. Proses Decompresi Algoritma LZW

Proses Decompresi pada LZW dilakukan dengan prinsip prinsip yang sama seperti proses kompresi. *dictionary* diinisialisasikan dengan semua karakter yang ada. Lalu pada setiap langkah, kode dibaca satu persatu dari *stream* code, di keluarkan *String* dari *dictionary* yang berkorespondensi dengan kode tersebut, dan di tambahkan *string* baru kedalam *dictionary* berikut algoritma *Decompresi* LZW.

Pada algoritma LZW proses *decompresi* tergantung kepada *dictionary* yang telah disusun pada saat proses kompresi, seperti pada contoh output yang di peroleh pada tabel V outputnya adalah : "1213310" dan telah diketahui data dalam *dictionary*, peroses decompresi LZW dapat dilihat pada gambar 2

Sehingga diperoleh hasil :



Gambar 2. Proses *Decompresi* LZW

D. Proses Kompresi Lempel Ziv Welch Fibonacci Search (LZWFS)

Pada dasarnya sistem kompresi LZW dan LZWFS memiliki algoritma yang sama, hanya saja pada proses pencarian data pada kamus data tidak diuji secara keseluruhan, melainkan menggunakan metode pancarian *Fibonacci Search* (FS), dapat dilihat Pada tabel VI. Penggabungan string 1 dan 2 pada langkah 5 yaitu "CA" Rumus Pencarian deret bilagan *Fibonacci Search* (FS) : f(n) = f(n-1) + f(n-2)

Tabel 6. Tahap Pencarian Ke-1 Dengan Algoritma Kompresi LZWFS

Dictionary	FIBO	FIBO
[1]A	F1 (tidak ditemukan)	-
[2]B	F2 (tidak ditemukan)	-
[3]C	F3(tidak ditemukan)	-
[4]D	-	-
[5] A B	F4(tidak ditemukan)	-
[6] B A	-	-
[7] A C	-	-
[8] C C	F5(tidak ditemukan)	F1 (Data tidak Ditemukan)
[9] C A		F2 (Data Ditemukan Pada index ke 9)

Pengujian pencarian nilai "CA" dilakukan sebanyak 7 langkah sesuai dengan algoritma *Fibonacci Search* (FS), Output Dari Proses Kompresi *Lempel Ziv Welch Fibonacci Search* (LZWFS) di atas Adalah "1213310"

E. Proses Decompresi Lempel Ziv Welch Fibonacci Search (LZWFS).

Pada proses *decompresi* algoritma *Lempel Ziv Welch Fibonacci Search* (LZWFS) pencarian data output pada dictionary dapat dilihat dibawah ini:
Output "1213310"

Pencarian langkah 1 dan 3: nilai output 1 dapat dilihat pada tabel 12

Tabel 7. Tahap Pencarian Langkah 1 dan 3. *Decompresi* LZWFS

Dictionary	FIB
[1]A	F1(data Ditemukan "A")
[2]B	-
[3]C	-
[4]D	-
[5] A B	-
[6] B A	-
[7] A C	-
[8] C C	-
[9] C A	-
[10]ACD	-

Pencarian langkah 2: nilai output 2 dapat dilihat pada tabel 8

Tabel 8. Tahap Pencarian langkah-2 *Decompresi* LZWFS

Dictionary	FIB
[1] A	F1(data tidak ditemukan)
[2] B	F2(data ditemukan "B")
[3] C	-
[4] D	-
[5] AB	-
[6] BA	-
[7] AC	-
[8] CC	-
[9] CA	-
[10]ACD	-

Pada tabel 3.8 akan di jelaskan tahap langkah ke-4 pencarian string ke dalam kamus data dengan algoritma LZWFS, data yang di cari adalah string "B" dan ditemukan di dalam dictionary pada index yang ke 2.
Pencarian langkah 4 dan 5: nilai output 3 dapat dilihat pada tabel 9.

Tabel 9. Tahap Pencarian langkah 4 dan 5 *Decompresi* LZWFS

Dictionary	FIB
[1]A	F1(data tidak ditemukan)
[2]B	F2(data tidak ditemukan)
[3]C	F3(data ditemukan "C")
[4]D	-
[5] A B	-
[6] B A	-
[7] A C	-
[8] C C	-
[9] C A	-
[10]ACD	-

Pada tabel 9 akan di jelaskan tahap langkah ke-4 pencarian string ke dalam kamus data dengan algoritma LZWFS, data yang di cari adalah string "C" dan ditemukan di dalam *dictionary* pada index yang ke 3.
Pencarian langkah 6: nilai output 10 dapat dilihat pada tabel 10.

Tabel 10. Tahap Pencarian langkah-6 *Decompresi* LZWFS

Dictionary	FIBO 1	FIBO 2
[1]A	F1(data tidak ditemukan)	-
[2]B	F2(data tidak ditemukan)	-
[3]C	F2(data tidak ditemukan)	-
[4]D	-	-
[5] A B	F2(data tidak ditemukan)	-
[6] B A	-	-
[7] A C	-	-
[8] C C	F2(data tidak ditemukan)	F1(data tidak ditemukan)
[9] C A	-	F2(data tidak ditemukan)
[10]ACD	-	F3(data ditemukan "ACD")

Pada tabel X di atas akan di jelaskan tahap langkah ke-4 pencarian string ke dalam kamus data dengan algoritma LZWFS, data yang di cari adalah string "ACD" dan ditemukan di dalam *dictionary* pada index yang ke 10. Sehingga dengan proses kompresi algoritma LZWFS yang telah dilakukan di peroleh hasil karakter sebagai berikut : "ABACCACD".

Tabel 11. Tampilan Hasil Kompresi LZW

Nama File	Ukuran File Asli (KB)	Ukuran File Kompresi (Kb)	Waktu Kompresi (S)	Rasio (Persen)	Proses
Test1.txt	6	3,55	26	40 %	100 %
Test2.txt	9	3,71	67	62,5 %	100 %
Test3.txt	12	4,07	98	63,64 %	100 %
Test4.txt	19	4,07	175	77,78 %	100 %
Test5.txt	24	5,34	228	78,27 %	100 %

Rasio = (Ukuran File Kompresi/Ukuran File Asli)*100%

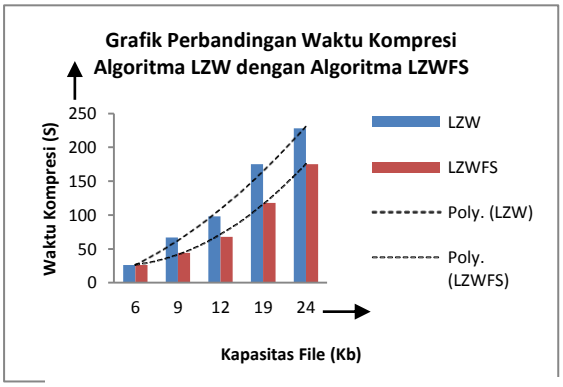
Hasil waktu dan ukuran file kompresi LZWFS dapat dilihat pada tabel 12

Tabel 12. Tampilan Hasil Kompresi LZWFS

Nama File	Ukuran File Asli (Kb)	Ukuran File Kompresi (Kb)	Waktu Kompresi (S)	Rasio (%)	Proses
Test1.txt	6	3,55	26	40 %	100 %
Test2.txt	9	4,92	44	50 %	100 %
Test3.txt	12	5,11	68	54,55 %	5,11
Test4.txt	19	5,12	118	72,22 %	5,12
Test5.txt	24	6,93	175	73,92 %	6,93

Tabel 13. Tabel Delta Waktu Kompresi

Nama File	LZW	LZWFS	Δ (Secon)
Teks 1	60	40	20
Teks 2	62,5	50	12
Teks 3	63,64	54,55	9,09
Teks 4	77,78	72,22	57
Teks 5	78,27	73,92	53



Gambar 3. Grafik Perbandingan Waktu Kompresi LZW dengan LZWFS

IV KESIMPULAN

A. *Kesimpulan*

Kompresi data berarti suatu teknik untuk memampatkan data agar diperoleh data dengan ukuran yang lebih kecil daripada ukuran aslinya sehingga lebih efisien dalam menyimpan serta mempersingkat waktu pertukaran data tersebut. Algoritma LZW ini dirancang cepat tetapi tidak bisa bekerja optimal karena hanya melakukan analisis terbatas pada data, namun algoritma LZW masih kurang optimal dalam mengkompres data.

Dari hasil penelitian dengan menguji lima jenis data teks menggukan algoritma LZW, dengan kapasitas yang berbeda yaitu 6,9,12,19 dan 24 Kb di peroleh rata – rata ukuran file kompresi sebesar 4, 148, rata-rata waktu kompresi sebesar 118,8, rata-rata rasio kompresi 64%. Dengan metode LZWFS di peroleh rata-rata ukuran file kompresi 5,126, rata-rata waktu kompresi 86,2, rata – rata rasio kompresi 58%. Dari hasil penelitian diatas di peroleh kesimpulan bahwa Algoritma LZWFS berhasil menyingkat waktu pencarian data namun masih memiliki kelemahan pada saat pengurutan data.

A. *Saran*

Dalam penelitian ini penulis telah berhasil melakukan pengoptimalan waktu kompresi dengan menggunakan metode LZWFS namun dalam metode LZWFS masih memerlukan waktu tambahan untuk mengurutkan data dalam kamus data, untuk itu penulis menyarankan kepada penulis selanjutnya agar mengembangkan metode pengurutkan data yang dapat digunakan untuk mengoptimlkan waktu kompresi dan di harapkan juga agar menggunakan pengujian yang lain.

REFERENSI

[1] Dzulhaq, M.I. & Andayani, A.A. 2014. Aplikasi Kompresi File Dengan Metode Lempel-Ziv-Welch. *Jurnal Sisfotek Global*, 4(1) : ISSN : 2088 – 1762.

[2] Neta, M.R.A. 2013. Perbandingan Algoritma Kompresi Terhadap Objek Citra Menggunakan JAVA. *Seminar Nasional Teknologi Informasi & Komunikasi Terapan* 979-26-0266-6.

[3] Kaur, D. & Kaur, K. 2013. Huffman Based LZW Lossless Image Compression Using Retinex Algorithm. *International Journal of Advanced Research in Computer and Communication Engineering* 2(8).

[4] Dzulhaq, M.I. & Andayani, A.A. 2014. Aplikasi Kompresi File Dengan Metode Lempel-Ziv-Welch. *Jurnal Sisfotek Global*, 4(1) : ISSN : 2088 – 1762.

[5] Neta, M.R.A. 2013. Perbandingan Algoritma Kompresi Terhadap Objek Citra Menggunakan JAVA. *Seminar Nasional Teknologi Informasi & Komunikasi Terapan* 979-26-0266-6.

[6] Jambek, A.B & Khairi, N.A, 2013. Performance Comparison Of Huffman And Lempel-Ziv Welch Data Compression For Wireless Sensor Node Application. *American Journal of Applied Sciences* 11(1) : 119-126.

[7] Smadi, M.A & Al-haji Q.A. 2014. A modified lempel–ziv welch source coding
Algorithm for efficient data compression. *Journal of Theoretical and Applied Information Technology* 61 (1) : 1817-3195.

[8] Kapoor, S., & Chopra, A. 2113. A Review of Lempel Ziv Compression Techniques.
International Journal Of Computer Science And Technology 1(4) : 0976-8491.

[9] Nishad, P. M, & Chezian, R.M 2014. Computational Complexity of Multiple Dictionary Lempel Ziv Welch (MDLZW) Algorithm and its Data Structure Implementations. *Australian Journal of Basic and Applied Sciences* 5(1).

[10] Taleb, S.A.A., Gharaybih, H.M.J., & Khtoom, A.A.M. 2010. Improving LZW Image
Compression. *European Journal of Scientific Research* 44(3) : ISSN 1450-216X V pp.
502-509.

[11] Nishad, P.M., & Chezian, R.M. 2012. optimization of lzw (lempel-ziv-welch) algorithm to reduce time complexity for dictionary creation in encoding and decoding Asian Journal Of Computer Science And Information Technology. *Asian Journal Of Computer Science And Information Technology* 2(5) : 114 – 118.

[12] Manikandan L.C & Selvakumar R.K. 2014. Interframe Video Coding using Fibonacci Spiral Block Matching Algorithm in H.264 Standard. *Australian Journal of Basic and Applied Sciences*, 8(18): 84-89.

[13] Singh, S., Yadav, P., & Mukherjee, G. 2015. Line Search Techniques by Fibonacci Search. *International Journal Of Mathematics And Statistics Invention* (IJMSI), ISSN: 2321 – 4767 P-ISSN: 2321 – 4759.

[14] Yahya, K. & Melita, Y. 2011. Aplikasi Kompresi Digital menggunakan teknik kompresi JPEG dengan Fungsi GUI pada matlab. *Jurnal Teknika* 3(2).

[15] Parthasarathy, C., Kalpana, G. & Gnanachandran, V. 2012. *LZW* Data Compression For Fsp Algorithm. *International Journal of Advanced Information Technology (IJAIT)* 2(5) : 631 - 561.

[16] Sayood, K. 2006. *Handbook Introduction to Data Compresion* ISBN-13: 978-0-12-620862-7.