OPTIMASI WAKTU DEKOMPRESI LOSSY MENGGUNAKAN METODE PENGELOMPOKAN JUMLAH-BIT KODE HUFFMAN PADA DATA LISA SATELIT LAPAN-A3
(TIME OPTIMIZATION FOR LOSSY DECOMPRESSION OF THE LISA SENSOR DATA ON LAPAN A3 SATELLITE USING A GROUPING METHOD OF HUFFMAN CODE BIT NUMBER)

Suhermanto[1], Rahmat Arief
Pusat Teknologi dan Data Penginderaan Jauh
Lembaga Penerbangan dan Antariksa Nasional
[1]e-mail: suhermanto@lapan.go.id
Diterima: 6 April 2018; Direvisi: 18 Juni 2018; Disetujui: 26 Juni 2018

ABSTRACT

The LAPAN-A3 satellite provides compressed multi spectral data from LISA sensor using real-time lossy compression. The LISA is an optical imaging sensor capable of imaging the Earth with 4 multispectral bands (blue, green, red and near-IR) and provides radiometric performance quantized over a 12-bit dynamic range. The lossy compression of the LISA multi spectral data is built from the Fourier transform and Huffman coding. A problem arised on the LISA data decompression process. It has taken about 12 hours for one scene of LISA data, which consist of 97120 scan lines. This happens because the searching method of the Hufman-code value runned sequentially from one bit to the next bit. In that case, this method is not ready to process large data. This paper proposed a new method of improvement of the LISA real-time lossy data decompression algorithm using clustering method of bit code on the the Huffman decoding algorithm and using pointers for data read and logic operations in buffer memory. The searching process of the clustered Huffman codes was done by tree diagram approach that starts from the smallest number of bits. The experiment was done using 6 LISA data samples. The result showed that the proposed method can speed up the processing time on average 15 times compared with the previous module. The compression ratio meet the LISA sensor design specification, i.e. less than 4 and the occurrence of the special character is very small i.e. less than 0.5%.

Keywords: *time optimization, lossy decompression, huffman code, LISA, LAPAN A3*

ABSTRAK

Satelit LAPAN-A3 menyediakan tiga pilihan untuk transmisi data multi-spektral LISA, yaitu tanpa kompresi, terkompres *lossy* ataupun terkompres *lossless*. Transmisi data multi-spektral menggunakan kompresi *real-time lossy*, dibangun menggunakan kombinasi transformasi Fourier dan enkoda-dekoda Huffman. Proses enkoda-dekoda Huffman data multi-spektral 4-kanal (biru, hijau, merah, dan *near infrared*) dengan resolusi radiometrik 12bit/*pixel* dikerjakan berbasis tabel statik dengan 514 kode biner. Permasalahan yang dihadapi saat dilakukan uji operasional modul dekompresi *lossy* adalah, kinerja modul sangat lambat dan diperlukan waktu cukup lama (hingga 12 jam) untuk mengolah 97120 baris data LISA atau setara dengan 185 detik pengamatan. Tulisan ini mengusulkan metode perbaikan algoritma dekompresi data LISA *real-time lossy* menggunakan pengelompokan jumlah-bit pada algoritma dekoda Huffman dan menggunakan *pointer* untuk pembacaan data dan operasi logika di memori buffer. Proses pencarian nilai-kode Huffman dilakukan menggunakan pendekatan diagram pohon yang dimulai dari jumlah-bit terkecil. Hasil uji kinerja pada 6 contoh data menunjukan bahwa modul dekompresi *lossy* yang diusulkan dapat mempercepat waktu proses rata-rata 15 kali dibandingkan dengan modul sebelumnya. Sementara itu, rasio kompresi *lossy* masih sesuai dengan spesifikasi desain yaitu 4 kali, dan persentase kemunculan data berkarakter khusus pada data tanpa cacat adalah sangat kecil yaitu kurang dari 0,5%.

Kata kunci: *optimasi waktu, dekompresi lossy, huffman code, LISA, LAPAN A3*

1 PENDAHULUAN

Pembuatan modul tampilan-cepat (*quicklook*) data satelit LAPAN-A3 telah dirintis sejak tahun 2016. Tahap pertama pengerjaannya diprioritaskan untuk menyelesaikan masalah ekstraksi dan tampilan data *SpaceCam* dan data instrumen *Line Imaging Space Application* (LISA) yang tidak dikompres. Kedua tipe data tersebut disediakan untuk mode akuisisi data diluar jangkauan antena penerima dan datanya disimpan di *onboard computer*. Ekstraksi kedua jenis data tersebut relatif sederhana karena operasinya didominasi oleh proses dekomposisi (Suhermanto, 2016). Waktu ekstraksi datanya relatif cepat, dan prosesor mampu menghasilkan tampilan-cepat mendekati *real-time* walau dieksekusi pada komputer desktop.

Selain terdapat mode transmisi data LISA tidak dikompres, satelit juga menyediakan mode transmisi data LISA terkompres. Pada akuisisi data *real-time* tersedia pilihan kompresi LISA *real-time lossy* atau LISA *real-time lossless* (LAPAN, 2015). Kedua jenis akuisisi *real-time* tersebut hanya dapat dioperasikan saat satelit berada dalam jangkauan antena penerima. Pada mode operasi ini, data hasil akuisisi instrumen LISA langsung dikompres, dipaket dan ditransmisikan ke stasiun Bumi penerima.

Penyandian data LISA *real-time lossy* dibangun menggunakan algoritma *Fast-Fourier Transform* (FFT) dan algoritma pengkodean entropi Huffman. Efisiensi kompresi sekitar 25%, tidak termasuk sandi yang ditambahkan pada saat pembentukan paket data *Consultative Committee for Space Data Systems* (CCSDS).

Ujicoba akuisisi data LISA secara *real-time* baru dilaksanakan awal tahun 2018, yaitu sejak beroperasinya stasiun bumi penerima data satelit LAPAN-A3/IPB di Rancabungur. Hasil uji kinerja perangkat lunak dekompresi LISA *real-time lossy* yang telah disiapkan ternyata sangat lambat dan tidak memenuhi

ekspektasi. Idealnya, waktu ekstraksi data mendekati waktu akuisisi agar kualitas data yang diakuisisi dapat dipantau melalui tampilan-cepat.

Dari dokumen evaluasi kinerja modul dekompresi *lossy*, untuk mendekompresi data pengamatan satu lintasan selama 185 detik atau setara 97120 baris data LISA, diperlukan waktu proses selama 12 jam. Padahal volume data yang diproses umumnya lebih besar bahkan dapat mencapai 170000 baris (Septi, Salaswati, dan Hakim, 2018).

Secara umum pengkodean entropi Huffman bekerja optimal pada data yang homogen, atau deviasinya kecil. Pengkodean ini menjadi tidak optimal pada data dengan deviasi besar karena berpeluang terdapat nilai piksel di luar tabel. Untuk menampung kode tersebut, disediakan kode/sandi berkarakter khusus. Penyandian berkarakter khusus memerlukan jumlah-bit lebih panjang yang dapat menurunkan efisiensi kompresi. Pada kondisi normal, kemunculan data berkarakter khusus sangat kecil. Akan tetapi, pada kondisi terjadi gangguan pada sinyal transmisi, kemunculan sandi berkarakter khusus dan gagal dekoda Huffman akan meningkat (Cormen, Leiserson, Rivest, dan Stein, 2009).

Beberapa penelitian mengenai penggunaan pengkodean Huffman pada citra satelit penginderaan jauh telah dilakukan. Swetha menerapkan pengkodean Run-Length and Huffman dalam JPEG 2000 pada data Landsat MSS untuk mendapatkan rasio kompresi 3.3 kali dan mendapatkan dekompresi citra dengan kualitas degradasi minimum (Swetha, John, dan Seetha Laksshmi, 2013). Sebuah metode kompresi citra multi spektral, hiperspektral dan LIDAR berbasis tranformasi *differential pulse code modulation* menggunakan pengodean entropi Huffman dilakukan oleh Ghamisi, Sepehrband, Kumar, dan Couceiro (2013) menghasilkan rerata rasio kompresi *lossless* lebih baik dari pada JPEG dan JPEG2000. Dalam penelitian lainnya, Chandhi mengimplementasikan teknik prediksi dan kode Huffman pada kompresi citra hiperspektral satelit AVIRIS (Chandhi dan Akhila, 2017). Proses enkoda-dekoda Huffman juga telah digunakan pada citra LAPAN-A3 menggunakan tabel Huffman statik. Walaupun menggunakan tabel statik, elemen tabel dapat diubah sesuai keperluan atau secara berkala. Pemakaian tabel statik cukup baik bila diterapkan pada satu atau beberapa citra tertentu, namun menghasilkan kinerja buruk bila diterapkan pada beberapa tipe citra berbeda (Hakim dan Permala, 2017). Metode pengelompokan jumlah-bit merupakan penyederhanaan dari beberapa algoritma dekoda Huffman, diantaranya penggunaan tabel *lookup* (Mansour, 2007), metode pengelompokan karakter untuk mengoptimasi penyandian Huffman (Gautam dan Murali, 2016), serta penggunaan teknik pencarian nilai-kode menggunakan model diagram pohon (Suri dan Goel, 2010).

Atas dasar pertimbangan kecepatan proses, makalah ini mengusulkan perbaikan algoritma dekompresi data LISA *real-time lossy* menggunakan metode pengelompokan jumlah-bit yang dikombinasi dengan penggunaan *pointer* dalam proses pencarian nilai kode maupun operasi logika di memori.

Besar kecilnya jumlah-bit penyusun tabel Huffman sangat tergantung pada fitur objek dan resolusi radiometrik data yang disandikan. Semakin homogen fitur objek, maka semakin sedikit pula jumlah-bit penyusun kode Huffman serta semakin

sederhana dan cepat proses *coding-decoding*. Hal ini disebabkan oleh mekanisme pencarian nilai-kode dimulai dari kode dengan jumlah-bit terkecil yang mewakili nilai-kode dengan probabilitas kemunculan paling tinggi.

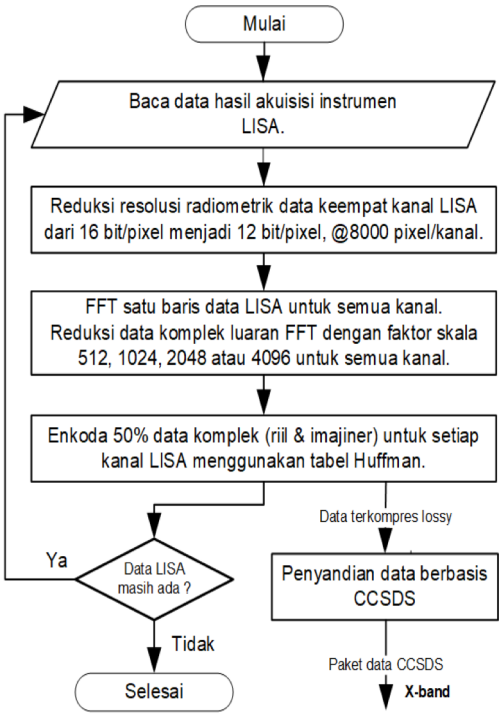
2 METODOLOGI

2.1 Kompresi paket data LISA *real-time lossy*

Data LISA *real-time lossy* tersusun atas 4 kanal spektral, 8000 piksel/baris, dan 12 bit/piksel. Alur penyandian data LISA *real-time* lossy di satelit disajikan pada Gambar 2-1. Kompresi *lossy* cukup banyak diterapkan pada transmisi data satelit karena memiliki keunggulan 1) dapat dilakukan secara *real-time*, dan 2) hasilnya tidak mengalami penurunan yang bermakna secara ilmiah (Beser, 1994). Namun penerapan kompresi pada transmisi data satelit lebih ditujukan untuk alasan kebutuhan *bandwidth* telemetri dan kebutuhan daya. Di samping itu, algoritma kompresi *lossy* juga digunakan untuk memperkecil kapasitas media penyimpanan data antara 3 hingga 20 kali.

Kompresi data LISA *real-time lossy* dilakukan baris demi baris. Proses enkoda satu baris data LISA dimulai dengan mereduksi resolusi radiometrik setiap kanal dari 16 bit menjadi 12 bit. Selanjutnya satu baris data LISA ditransformasi Fourier (FFT), sehingga diperoleh 4 set data komplek (riil dan imajiner) yang masing masing terdiri atas 8192 elemen. Nilai elemen komplek tersebut cukup besar dan harus di skala terlebih dahulu sebelum diumpankan ke enkoda Huffman. Tersedia empat faktor skala, yaitu 512, 1024, 2048 dan 4096.

Enkoda Huffman hanya diterapkan pada 50% data komplek saja. Dari total 8092 elemen komplek, enkoda hanya dilakukan pada 4097 elemen riil dan 4097 elemen imajiner. Hasil enkoda ke empat kanal data LISA selanjutnya diumpankan ke penyandian data berbasis CCSDS.



Gambar 2-1: Bagan penyandian data LISA *real-time lossy* secara global

Paket data hasil enkoda disusun baris demi baris dengan memberi kode pemisah antara kelompok (blok) data dari setiap kanal, maupun antara data riil dan data imajiner walau dari kanal yang sama. Penulisan satu blok data dimulai dengan START MARKER dan diakhiri dengan STOP MARKER. Selanjutnya terhadap setiap baris data LISA terkompres lossy ditambahkan informasi nomor baris dan *time-stamp* untuk kemudian satu baris data terkompres tersebut dimasukkan ke struktur *Virtual Channel Data Unit* (VCDU) pada struktur CCSDS untuk ditransmisikan ke Bumi. Perihal penyandian data LISA berbasis CCSDS tidak dibahas pada tulisan ini.

2.2 Dekompresi paket data LISA *real-time lossy*

Diagram alir dekompresi LISA *real-time lossy* ditampilkan pada Gambar 2-2.

Data masukan pada penelitian ini adalah data mentah (*raw-data)* LISA luaran perangkat *High Rate Data Modem* (HDRM) yang telah didekoda CCSDS (antara lain, derandomisasi dan Reed-Solomon), tetapi hasilnya tetap disimpan dalam format CCSDS (Suhermanto, 2016). Proses dekompresi *lossy* dimulai dengan mengkonversi data mentah menjadi paket ilmiah, yaitu data LISA *lossy* yang telah terbebas dari semua artefak komunikasi. Paket ilmiah LISA memuat sejumlah baris data LISA terkompres *lossy* dengan volume yang bervariasi tergantung pada periode perekaman.

Proses dekoda Huffman dikerjakan berbasis tabel statik yang berisi kode, mewakili 514 nilai-kode (karakter). Nilai-kode berisi data dengan nilai mulai dari -256 s/d +256, ditambah satu nilai untuk menampung nilai-kode khusus (*special character*). Kode khusus adalah satu diantara kode dalam tabel huffman dengan panjang 14 bit yang memiliki tiga makna. Kemunculan kode khusus selalu diikuti dengan kode lanjutan yang terdiri atas 17 bit. Bila 17 bit kode lanjutan tersebut bernilai nol ini berarti tanda START MARKER, sedang jika bernilai satu berarti STOP MARKER. Adapun selain nol dan satu berarti memuat nilai-kode yang tidak dapat ditampung didalam tabel. Dalam hal ini nilai-kode tersebut < -256 atau > 256 tetapi tetap dalam cakupan nilai 12 bit.

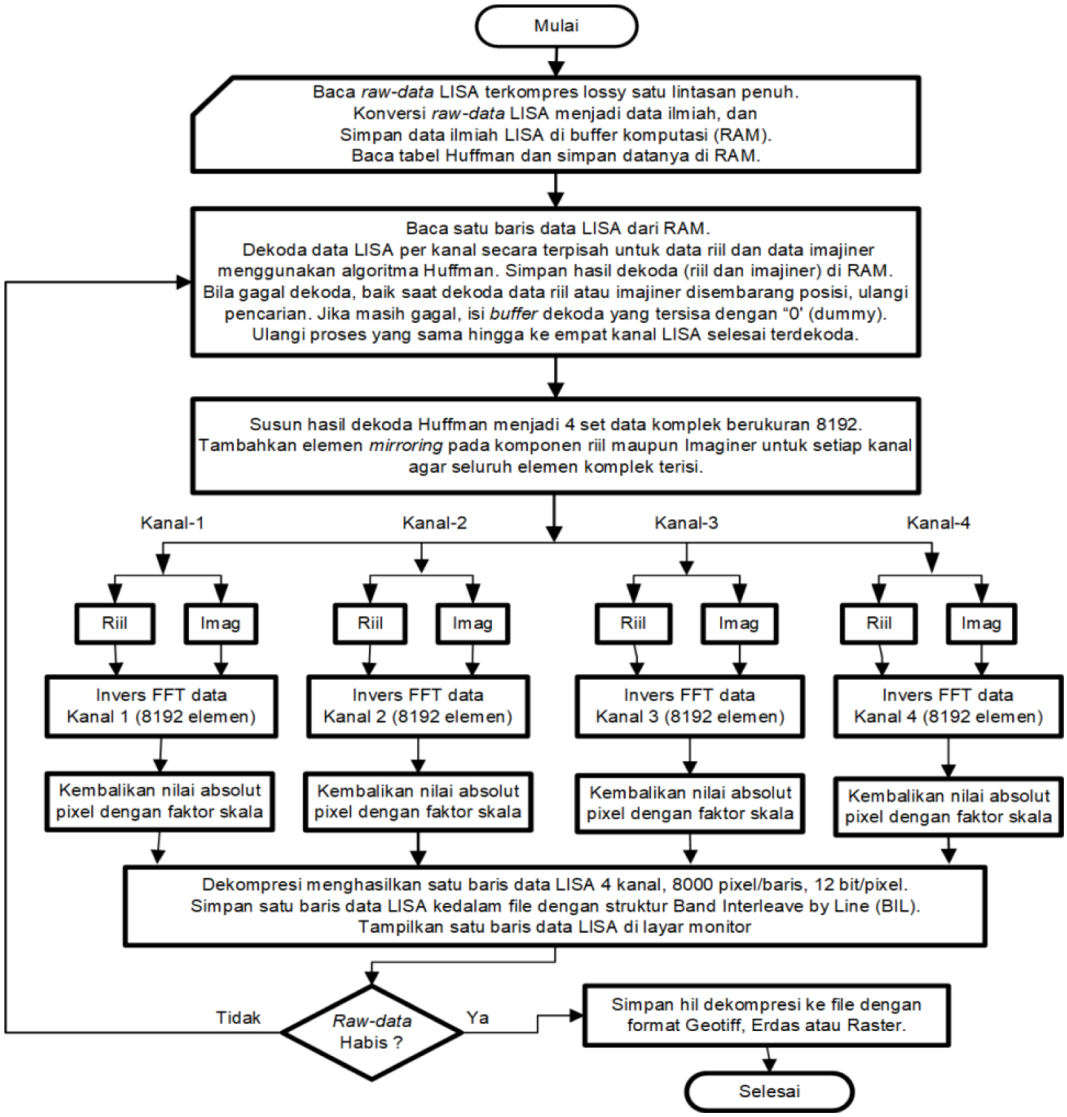
Sesuai alur enkoda data LISA, proses dekoda juga dilakukan baris demi baris. Satu baris data terkompres didekoda delapan kali, 4 kali untuk mendapatkan data riil dan 4 lainnya untuk data imajiner. Data tersebut kemudian disusun menjadi 4 set data komplek dengan masing masing terdiri atas 4097 elemen. Selanjutnya, *mirroring* dilakukan terhadap setiap kanal data tersebut sehingga setiap set data komplek terdiri atas 8192 elemen. Satu set data komplek mewakili satu kanal. Setelah melakukan *inverse* FFT terhadap 4 set data komplek, selanjutnya akan diperoleh satu baris data LISA yang terdiri atas 4 kanal. Satu baris data ini kemudian ditampilkan dan disimpan ke *file* sesuai format data yang diminta.

Pendekoda Huffman sangat peka terhadap kesalahan bit. Bilamana setelah pencarian hingga bit ke-15 (bit terakhir) kode bit tidak juga ditemukan, tidak ada acuan yang dapat digunakan untuk mengetahui posisi bit yang salah. Upaya yang mungkin dilakukan adalah diasumsikan terjadi *slip-bit* dalam blok data tersebut. Dengan asumsi tersebut, pencarian ulang akan dilakukan dengan memulainya dari posisi yang semestinya dengan 1 bit maju kemudian dicari dan 1 bit mundur. Jika upaya ini gagal, semua nilai-kode yang belum diisi pada blok tersebut akan diisi dengan nilai nol (*dummy*).

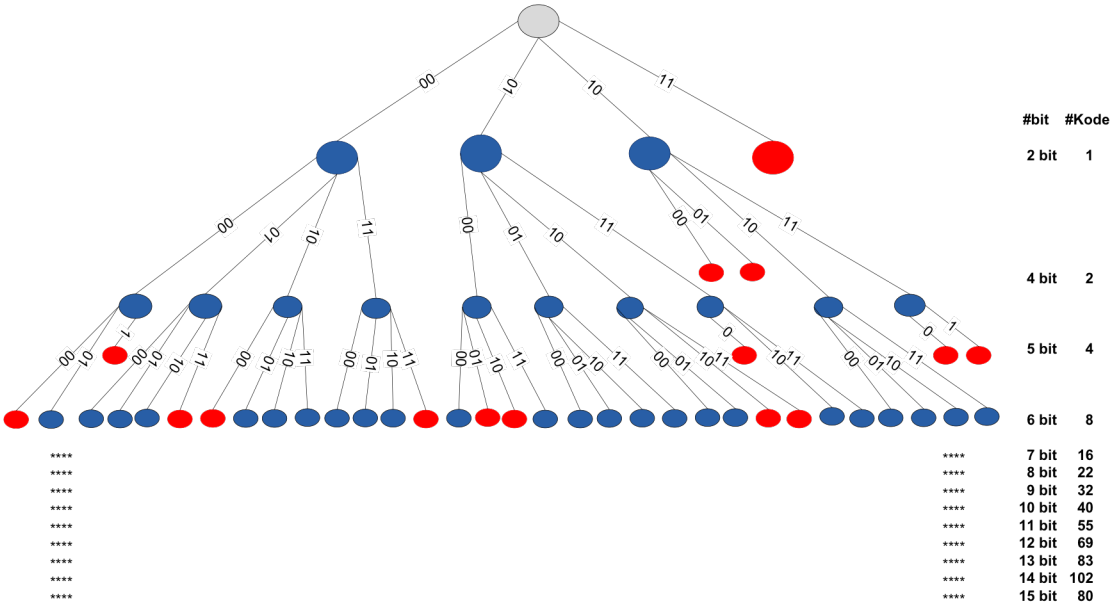
2.3 Algoritma dekoda Huffman

Prinsip dekoda Huffman ditampilkan pada diagram pohon dalam Gambar 2-3. Bagan tersebut hanya menampilkan diagram Huffman hingga jumlah-bit=6 dari seharusnya hingga jumlah-bit=15. Besar kecilnya jumlah-bit penyusun tabel Huffman tergantung pada fitur objek, resolusi radiometrik serta jumlah kanal data yang disandikan. Semakin homogen fitur objek jumlah-bit kodenya juga semakin sedikit dan proses dekoda datanya semakin sederhana dan cepat.

Pencarian nilai-kode dimulai dari kode dengan jumlah-bit terkecil dengan target menemukan *node* warna merah (Gambar 2-3). *Node* warna merah adalah *leaf node* yang berisi nilai-kode Huffman.



Gambar 2-2: Alur proses dekompresi data LISA terkompres *lossy*.



Gambar 2-3: Diagram pohon tabel Huffman pada penyandian data LISA *lossy*

```
// Sub-program dekoda Huffman dengan metode pengelompokan jumlah-bit kode Huffman.
// Pencarian nilai-kode dimulai dari kode 2-bit  hingga 15 bit.
// Susunan bit kode yang ingin diketahui tersimpan didalam variable BinCode
//
        int DecodeDataA3::DekomposisiDataHuffLossy()
        {
        int             i, j, LVal = 0;
        BinCode = 0;STAT = 1;         // Set parameter awal pencarian

        for (i = 2; i <16; i++) {                       // Set jumlah pencarian kode Huffman
          if (i == 2)   BinCode = read2bit(0);       // Inisialisasi buffer BinCode, dimulai dengan pembacaan 2 bit
           else {
              BinCode <<= 1;
                if (read1bit()) BinCode += 1;       // Tambahkan satu bit kode berikutnya
           }
          switch (i) {                                  // Proses pencarian nilai-kode Huffman
                case 2:  if (BinCode == LY_Code[0]) return(LY_XVal[0]);
                         break;
                case 3:             break;
                case 4:  for (j = 1; j<3; j++) { if (BinCode == LY_Code[j]) return(LY_XVal[j]); }
                         break;
                case 5:  for (j = 3; j<7; j++) { if (BinCode == LY_Code[j]) return(LY_XVal[j]); }
                         break;
                case 6:  for (j = 7; j<15; j++) { if (BinCode == LY_Code[j]) return(LY_XVal[j]); }
                         break;
                case 7:  for (j = 15; j<31; j++) { if (BinCode == LY_Code[j]) return(LY_XVal[j]); }
                         break;
                case 8:  for (j = 31; j<53; j++) { if (BinCode == LY_Code[j]) return(LY_XVal[j]); }
                         break;
                case 9:  for (j = 53; j<85; j++) { if (BinCode == LY_Code[j]) return(LY_XVal[j]); }
                         break;
                case 10:  for (j = 85; j<125; j++) { if (BinCode == LY_Code[j]) return(LY_XVal[j]); }
                         break;
                case 11:  for (j = 125; j<180; j++) { if (BinCode == LY_Code[j]) return(LY_XVal[j]); }
                         break;
                case 12:  for (j = 180; j<249; j++) { if (BinCode == LY_Code[j]) return(LY_XVal[j]); }
                         break;
                case 13:  for (j = 249; j<332; j++) { if (BinCode == LY_Code[j]) return(LY_XVal[j]); }
                         break;
                case 14:  for (j = 332; j<433; j++) { if (BinCode == LY_Code[j]) return(LY_XVal[j]); }
                         if (BinCode == 5451) {
                             LVal = read17bit();                        // SPECIAL CHARACTER VALUE
                              if (LVal == 0) { STAT = 4; return (LVal); }        // DATA START MARKER
                                else {
                                  if (LVal == 1) { STAT = 5; return(LVal); }     // DATA END MARKER
                                    else { STAT = 2; return(LVal); }                // Data pixel dengan nilai khusus
                         }
                }
                break;
                case  15:   for (j = 433; j<513; j++) { if (BinCode == LY_Code[j]) return(LY_XVal[j]); }
                                STAT = 0;   // Kode Huffman tidak ditemukan ➔ Pencarian gagal
                         break;
                default:  STAT = 0; break;      // Kode Huffman tidak ditemukan ➔ Pencarian gagal
                }
          }
          return (0);                   // Return status ➔ Error
        }
```

Gambar 2-4: Sub-program dekoda Huffman berdasarkan pengelompokan jumlah-bit

Bila *leaf node* belum ditemukan, pencarian dilanjutkan pada cabang berikutnya dengan menambah satu bit kode. Proses pembacaan, pergeseran bit dan perbandingan nilai kode menjadi aktifitas terbesar dalam mencari satu nilai kode Huffman. Pencarian berhenti bila ditemukan nilai-kode atau kode tidak ditemukan hingga bit ke-15. Jika hingga hingga bit ke-15 kode tidak ditemukan, pencarian akan diulang. Bilamana gagal, proses dihentikan dan diberi status bahwa pencarian gagal.

Pada Gambar 2-3, terdapat dua *node* yang tidak memiliki *leaf node*, yaitu untuk jumlah - bit = 1 dan 3. Sehingga hanya ada 13 sub-pencarian kode. Implementasi algoritma Huffman pada

modul dekompresi LISA *real-time* lossy merupakan penyederhanaan dari metode pendekoda Huffman yang menggunakan *table lookup* dengan metode pencarian kode berdasarkan pengelompokan jumlah-bit (Gambar 2-4). Sedangkan teknik *pointer* digunakan untuk mempercepat proses dan efisiensi dalam pembacaan data dan operasi logika. Teknik ini dapat mengakses data langsung ke memori komputer (Hong-Chung, Yue-Li, dan Yu-Feng, 1999).

2.4 Dekompresi data menggunakan FFT

Penggunaan FFT untuk kompresi data/citra sangat bervariasi. Alfalou memperkenalkan metode kompresi *lossy* spektral citra yang dapat mereduksi penggunaan memori dan dapat menghasilkan citra asli secara adaptif hanya dengan menggunakan informasi *phase* (Alfalou, Elbouz, Mansour, dan Keryer, 2010). Penggunaan FFT dikombinasikan dengan kuantisasi skalar dan penyandian Huffman juga telah diimplementasikan di satelit (Sahnoun dan Benabadji, 2014). Di sisi lain, penggunaan *Fast Fourier coefficient Transform* pada blok matrik berbagai ukuran yang tidak tumpang-tindih (*overlapping*) telah diuji pada berbagai citra (Reddy dan Rani, 2016).

Implementasi *Invers* FFT (IFFT) pada dekompresi data LISA lossy hanya menggunakan 50% dari data asli, tentunya masih lebih besar dibandingkan dengan metode yang ditawarkan Alfalou. Walau keduanya menggunakan teknik *mirroring* untuk mengisi elemen IFFT yang kosong, tetapi pada dekompresi data LISA tidak perlu melakukan proses iterasi.

3 HASIL DAN PEMBAHASAN

Algoritma kompresi Huffman sangat rentan terhadap kesalahan data, tidak terkecuali kesalahannya hanya satu bit. Kesalahan bit data berpeluang terjadi saat transmisi maupun saat proses modulasi atau demodulasi.

Walau penyandian data LISA telah mengikuti rekomendasi CCSDS, namun masih terdapat sejumlah blok data yang gagal dekoda dari hasil dekoda Huffman. Sesungguhnya penyandian CCSDS telah sangat baik melindungi data dengan meningkatkan kehandalan/keamanan data selama transmisi melalui penerapan *codec* Reed-Solomon dan randomisasi data. Akan tetapi, penerapan *codec* tersebut terbatas pada setiap 223 *byte* (*codeword*). Akibatnya, walau kode START MARKER dan STOP MARKER terdeteksi dengan baik, hal ini tidak menjamin bahwa hasil akan terbebas dari kesalahan.

Gangguan interferensi sinyal adalah kendala utama pada penerimaan data satelit di pita-X. Hal ini dikarenakan regulasi pemerintah Indonesia yang menempatkan kegiatan Satelit Eksplorasi Bumi (angkasa ke Bumi) dan sistem komunikasi terrestrial dari titik ke titik memiliki kedudukan yang sama yaitu sama-sama primer pada pita tersebut (Kemkominfo RI, 2014). Oleh sebab itu, interferensi oleh perangkat komunikasi terrestrial, khususnya komunikasi antar BTS, menjadi salah satu sumber gangguan pada arah tertentu.

Kesalahan dekompresi dapat juga terjadi pada data saat akuisisi dengan sudut elevasi antena rendah. Fenomena ini umum terjadi pada awal dan akhir proses akuisisi. Umumnya, akuisisi data pada elevasi rendah sangat dipengaruhi oleh *ground noise*.

Wujud gangguan pada citra yang diakibatkan oleh kedua sumber gangguan, interferensi dan *ground noise,* sangat sulit dibedakan. Hal ini hanya mungkin diketahui bila analisanya

dilengkapi dengan data posisi antena untuk dikorelasikan dengan sudut elevasi antena dan sumber gangguan. Gambar 3-1b adalah cuplikan citra dengan cacat yang disebabkan oleh *ground noise* atau akuisisi data pada elevasi rendah. Sedangkan pada Gambar 3-1a, ditampilkan contoh citra yang secara visual tidak mengalami distorsi.

Upaya koreksi nilai-kode yang gagal dekoda telah dilakukan melalui pencarian ulang, namun tidak berhasil. Salah satu penyebabnya adalah karena satu baris data LISA terkompres *lossy* yang rata-rata berukuran 16000 *byte* disimpan dalam beberapa *Channel Access Data Unit* (CADU) (CCSDS, 2017). Berdasarkan penyandian data LISA, satu baris data akan disimpan dalam ±70 *codeword*, dimana satu CADU terdiri atas 5 *codeword* (Suhermanto, 2016). Kemungkian terjadi kesalahan *codeword* dalam satu baris data LISA pada data data yang mengalami gangguan adalah sangat tinggi. Hal ini menjadi penyebab dekoda gagal walaupun parameter yang terkait dengan keperluan dekoda data dapat diekstraksi dengan baik.

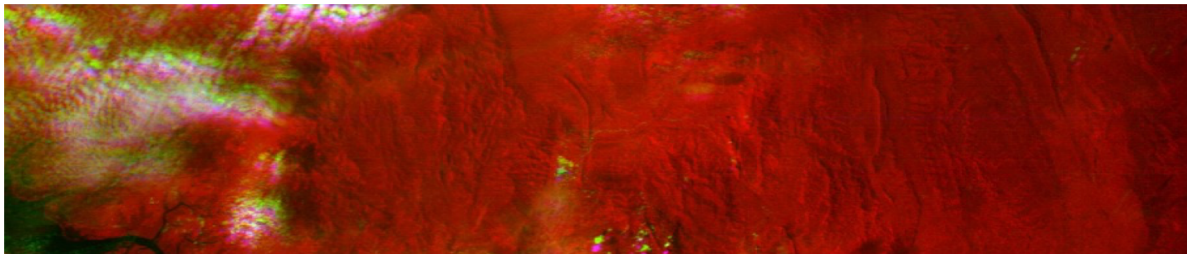
Kecepatan dekoda Huffman akan menurun bila ditemukan kesalahan kode biner. Pencarian data berulang dapat berlanjut hingga kehilangan satu blok data atau bahkan kehilangan satu kanal data dan terus hingga kehilangan satu baris. Bila sinyal gangguan sampai merusak kode START MARKER *lossy*, maka satu baris data LISA akan hilang dan hal ini dapat diketahui dari data *timestamp* yang menyertainya. Hal yang sama terjadi bila gangguan menyebabkan kode kompresi *lossy* yang tidak terdeteksi, maka satu baris data juga akan hilang.

Rasio kompresi FFT pada data LISA *real-time lossy* hanya 50%, namun Alfalou dalam publikasinya (Alfalou et al., 2010) memperkenalkan kompresi FFT dengan rasio hingga 25%. Metode yang dikemukakan hanya mengirimkan 50% data imajiner tanpa mengirim data riilnya. Namun, keunggulan rasio kompresi harus dikompensasi dengan melakukan proses iterasi *invers* FFT. Metode ini tidak mungkin dilakukan pada data LISA karena volumenya sangat besar dan proses invers FFT dilakukan baris demi baris.

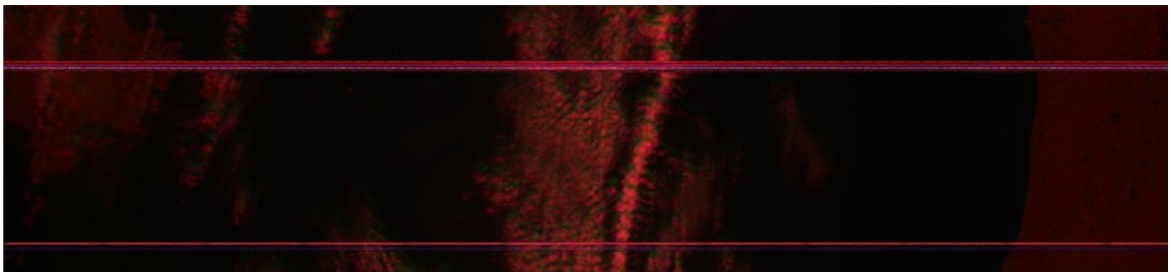
Pembentukan data komplek pada algoritma Alfalou dilakukan dengan mengisi semua elemen riil dengan 1, sementara kekurangan data pada elemen imajiner diisi dengan *mirroring* datanya. Berbeda dengan proses yang diterapkan pada pembentukan data komplek LISA, *mirroring* data komplek dilakukan pada komponen riil dan imajiner, namun tanpa proses iterasi IFFT.

Hasil uji kinerja modul dekompresi LISA *real-time lossy* dirangkum pada Tabel 3-1. Secara umum metode yang dikembangkan berhasil mengoptimasi waktu proses secara signifikan. Merujuk pada waktu dekompresi data LA3_20180130_030446.rso, dari yang semula sekitar 12 jam berhasil ditingkatkan menjadi sekitar 47,58 menit, sehingga rata-rata metode yang diterapkan mampu mempercepat proses hingga 15,13 kali.

Berdasarkan data pada Tabel 3-1, rasio kompresi data *real-time lossy* rata-rata lebih dari 4 kali. Nilai ini sesuai dengan spesifikasi desain yang mengharapkan rasio kompresi 4 kali, walau perhitungannya masih berdasarkan pada data ilmiah yang di dalamnya masih terdapat kode lain. Data ini sekaligus membuktikan bahwa tabel Huffman statik yang disiapkan sejak sebelum peluncuran masih relevan digunakan hingga saat ini.



Gambar 3-1a: Tampil cuplik data LA3_20180129_014812_S.rso yang diakuisisi tanggal 29/01/2018 Jam 08:48:12



Gambar 3-1b: Tampil cuplik data LA3_20180130_030446_S.rso yang diakuisisi tanggal 30/01/2018 Jam 10:04:46

Tabel 3-1: RANGKUMAN PROSES DEKOMPRESI DATA LISA *REAL-TIME* LOSSY MENGGUNAKAN METODE PENGELOMPOKAN JUMLAH-BIT

Nama file	Volume Data input (Kbyte)	Volume Data Ilmiah (Kbyte)	Jumlah baris LISA (scanline)	Volume hasil dekompresi (Kbyte)	Waktu proses (menit)	Rasio kom-presi *)
LA3_20180129_014812.rso	960.041	618.457	43.005	2.687.875	19,767	4,346
LA3_20180130_030446.rso	2.381.620	1.284.399	97.137	6.071.125	47,583	4,727
LA3_20180201_022924.rso	2.520.755	1.775.995	115.736	7.233.563	52,483	**4.073**
LA3_20180215_030808.rso	1.873.095	1.109.931	85.387	5.336.750	41,867	4,808
LA3_20180216_025017.rso	584.575	408.731	26.857	1.678.625	12,500	**4.107**
LA3_20180217_023239.rso	3.702.203	2.350.139	169.385	10.586.625	82,067	4,505

*) Dihitung terhadap data ilmiah yang masih memuat kode START MARKER dan STOP MARKER

Tabel 3-2: PERSENTASE KEMUNCULAN DATA BERKARAKTER KHUSUS DAN JUMLAH BLOK DATA RIIL/IMAJINER YANG GAGAL DEKODA

Nama file	Jumlah baris (scanline)	Data dengan nilai khusus	Decode Huffman dengan Blok *Error*	% data dengan karakter khusus *)	% jumlah baris dgn blok data salah **)
LA3_20180129_014812_S.rso	2.141	34.413	159	0.0502	0,928
LA3_20180130_030446_S.rso	2.249	125.505	492	0,1743	2,734
LA3_20180129_014812.rso	43.005	1.526.323	5.278	0,1109	1,534
LA3_20180130_030446.rso	97.137	3.735.399	13.005	0,1202	1,673
LA3_20180201_022924.rso	115.736	**11.678.795**	34.969	**0,3153**	**3,777**
LA3_20180215_030808.rso	85.387	4.164.320	12.936	0,1524	1,894
LA3_20180216_025017.rso	26.857	**3.330.281**	12.587	**0,3875**	**5,858**
LA3_20180217_023239.rso	169.385	6.857.791	23.146	0,1265	1,708

*) Persentase dihitung terhadap total pixel pada data bersangkutan.
**) Tidak memperhitungkan kehilangan baris data karena kode Lossy tidak ditemukan.

Dari catatan proses dekompresi data LISA *lossy,* diketahui cukup banyak blok data yang tidak dapat diidentifikasi pada semua sampel yang diuji, termasuk data sampel yang secara visual tidak terlihat adanya cacat. Persentasi kesalahan tersebut disajikan pada Tabel 3-2. Bila diperhatikan lebih lanjut, Gambar 3-1a yang secara visual tidak terlihat adanya cacat, ternyata terdeteksi 159 blok data yang tidak berhasil di dekoda (data LA3_20180129_014812_S.rso pada Tabel 3-2. Selain itu, terdapat 34.413 piksel yang disandikan berkarakter khusus. Hal Ini menunjukkan bahwa cuplikan data tersebut memiliki 0,05% data piksel yang nilainya tidak tercakup dalam tabel Huffman.

Bila dibandingkan dengan data LA3_20180130_030446_S.rso yang memiliki jumlah baris hampir sama dengan data sebelumnya namun memiliki sejumlah cacat baris (Gambar 3-1b), ternyata populasi data berkarakter khusus dan gagal dekoda meningkat. Dari Tabel 3-2 diketahui bahwa terjadi peningkatan populasi gagal dekoda yang signifikan. Hal ini menunjukkan bahwa gangguan akibat *ground noise* maupun *link* transmisi data komunikasi akan menghasilkan pengaruh yang sama, yaitu peningkatan populasi kedua parameter tersebut. Dalam hal ini peningkatan populasi data berkarakter khusus bukan berasal dari citra, tetapi muncul dari data acak yang bersesuaian dengan kode karakter khusus.

Terdapat korelasi antara rendahnya rasio kompresi pada sampel ketiga dan kelima di Tabel 3-1 dengan banyaknya data berkarakter khusus pada Tabel 3-2. Hal ini dikarenakan setiap kemunculan satu nilai-kode berkarakter khusus perlu 31 bit, dari normalnya kurang dari 15 bit. Kemunculan data berkarakter khusus tidak dapat dihindari, namun dapat diperkecil dengan memperbaharui tabel Huffman. Dari data Tabel 3-2, populasi data berkarakter khusus masih di bawah 0,2% pada data tanpa cacat dan kurang dari 0,4% pada data yang mengalami gangguan. Oleh sebab itu, efektivitas tabel Huffman statik masih memadai dan tidak perlu dilakukan pembaharuan tabel.

4 KESIMPULAN

Algoritma dekompresi LISA *real-time lossy* menggunakan metode yang diusulkan berhasil dikembangkan dan telah diuji menggunakan 6 sampel dengan periode akuisisi Januari-Februari 2018. Kinerja modul telah teruji untuk data berukuran mulai dari 580MByte hingga 3,7Gbyte, dengan perbaikan waktu proses rata-rata 15 kali dibandingkan dengan metode sebelumnya.

Persentase data berkarakter khusus pada sampel data tanpa gangguan menggunakan tabel Huffman statik dengan 514 kode nilainya masih sangat rendah (<0,5%). Ini menunjukkan bahwa tabel Huffman statik masih relevan digunakan hingga kini. Akan tetapi gangguan terhadap *link* komunikasi dari satelit ke stasiun penerima di Bumi dapat terganggu dan menghasilkan aliran data acak. Aliran data acak akan meningkatkan persentase gagal dekoda maupun data berkarakter khusus.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Kepala Pusat Teknologi Satelit dan Kepala Pusat Teknologi dan Data Penginderaan Jauh, LAPAN, atas dukungan sarana dan fasilitas. Juga terhadap berbagai masukan dari *blind reviewer* dalam rangka membuat tulisan ini menjadi lebih baik, penulis ucapkan terima kasih.

DAFTAR RUJUKAN

Alfalou, A., Elbouz, M., Mansour, A., and Keryer, G. (2010). New spectral image compression method based on an optimal phase coding and the RMS duration principle. *Journal of Optics*, *12*(11). https://doi.org/10.1088/2040-8978/12/11/115403

Beser, N. D. (1994). Space Data Compression Standards. *Johns Hopkins APL Technical Digest*, *15*(3), 206–209.

CCSDS. (2017). CCSDS 131.0-B-3 Recommendation for Space Data System Standards; TM Synchronization and Channel Coding. *Recommendation for Space Data System Standards*.

Chandhi, R., and Akhila, S. (2017). Hyperspectral Image Compression using Huffman Coding and Prediction Technique. *International Journal of Electrical, Electronics and Data Communication*, *5*(8), 67–71.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms, Third Edition* (3rd ed.). The MIT Press.

Gautam, R., and Murali, S. (2016). An Optimized Huffman's Coding by the method of Grouping. *ArXiv:1607.08433*. Retrieved from http://arxiv.org/abs/1607.08433

Ghamisi, P., Sepehrband, F., Kumar, L., and Couceiro, M. S. (2013). A new method for compression of remote sensing images based on an enhanced differential pulse code modulation transformation. *ScienceAsia*, *39*(5), 546–555. https://doi.org/10.2306/scienceasia1513-1874.2013.39.546

Hakim, P. R., dan Permala, R. (2017). Analysis of LAPAN-IPB image lossless compression using differential pulse code modulation and huffman coding. *IOP Conference Series : Earth and Environmental Science*, *54*(1),1–9. https://doi.org/10.1088/1755-1315/54/1/012096

Hong-Chung, C., Yue-Li, W., and Yu-Feng, L. (1999). A memory-efficient and fast Huffman decoding algorithm. *Information Processing Letters*, *69*(3), 119–122. https://doi.org/10.1109/AINA.2005.33

Kemkominfo RI. (2014). PERMENKOMINFO RI no 25 Tahun 2014 Tentang Tabel Alokasi Spektrum Frekuensi Radio Indonesia. Jakarta: KOMINFO.

LAPAN, P. T. S. (2015). LAPAN-A3/IPB Polar Microsatellite for Remote Sensing Experiment. Detail Design. Doc: LA3-DD-2015-04. Bogor: Pusat Teknologi Satelit.

Mansour, M. F. (2007). Efficient Huffman Decoding With Table LookUp. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on* (pp. 53–56). IEEE.

Reddy, M. K., and Rani, S. A. J. (2016). Statistical Image Compression using Fast Fourier Coefficients, *155*(3), 31–36.

Sahnoun, K., and Benabadji, N. (2014). on-Board Satellite Image Compression Using the Fourier Transform and Huffman Coding. *The International Journal of Computational Science, Information Technology and Control Engineering*, *1*(1), 17–23.

Septi, A. P., Salaswati, S., dan Hakim, P. R. (2018). *Technical Note TN.5.2.3 WP 5.2 Analisis Payload - Pengolahan Data Citra Kamera Multispektral LAPAN-A3 dalam Mode Operasi Realtime*. Bogor.

Suhermanto. (2016). PENGUJIAN MODUL PENGOLAH DATA TELEMETRI LAPAN-A3 / IPB UNTUK MENGHASILKAN PRODUK LEVEL-0 (THE TESTOF LAPAN-A3 / IPB TELEMETRY DATA PROCESSOR MODULE TO PRODUCE LEVEL-0 PRODUCT). *Teknologi Dirgantara*, *14*(2), 125–136.

Suri, P. R., and Goel, M. (2010). Ternary Tree and A New Huffman Decoding Technique, *10*(3), 165–172.

Swetha, V., John, J. P., and Seetha Laksshmi, T. . (2013). Data Compression of Remote Sensing Images using Wavelet Transforms. *International Journal of Recent Advances in Engineering and Technology (IJRAET)*, *1*(2), 21–26.