TRANSMISI CITRA MENGGUNAKAN JOINT LDPC DECODING

Aries Pratiarso[1], M. Agus Zainuddin[2]
Jurusan Teknik Telekomunikasi Politeknik Elektronika Negeri Surabaya
Institut Teknologi Sepuluh Nopember Surabaya
Kampus ITS Sukolilo Surabaya 60111
Telp. (031) 5947280 ext. 4109, Faks. (031) 5946114
e-mail: aries@eepis-its.edu, magusz@eepis-its.edu

ABSTRAK

Metode transmisi citra menggunakan teknik Joint LDPC decoding merupakan teknik untuk mengirimkan citra menggunakan teknik diversitas dan pengkodean pengkoreksi kesalahan. Pada sisi pengirim, citra dikompresi menggunakan algoritma SPIHT. Data citra terkompresi kemudian dikirimkan melalui kanal paralel. Hal ini ditujukan bila pada salah satu kanal terjadi gangguan dan data tidak bisa didekodekan, maka data masih bisa diperoleh dari kanal lainnya. Hasil simulasi menunjukkan teknik joint LDPC decoding pada transmisi citra dapat meningkatkan kualitas citra rekonstruksi yang mampu memberikan coding gain sebesar 1,3 dB untuk BER 10-4 terhadap single LDPC decoding. Penggunaan teknik joint LDPC decoding secara rata-rata memberikan peningkatan PSNR sebesar 4,44 dB. Hasil tersebut menunjukkan bahwa metode yang digunakan dapat meningkatkan performansi sistem transmisi citra.

Kata Kunci: Joint LDPC decoding, kompresi citra SPIHT, kode LDPC, iterative decoding

1. PENDAHULUAN

Pada dekade terakhir ini terjadi peningkatan kebutuhan komunikasi data, baik dari segi layanan, kehandalan sistem, maupun laju transmisinya. Dengan berintegrasinya teknologi *wireless* dan layanan multimedia, pentransmisian citra dan video dengan kualitas yang baik, menjadi satu dari tujuan dari sistem jaringan *wireless* generasi selanjutnya (4G dan seterusnya).

Sinyal multimedia memiliki ukuran data yang besar, sehingga dibutuhkan teknik kompresi apabila akan ditransmisikan ataupun disimpan dalam media penyimpan data. SPIHT dan JPEG 2000 merupakan algoritma kompresi citra yang mampu mencapai rasio kompresi yang tinggi. Namun aliran data terkompresi sangat rentan terhadap gangguan kanal, meski untuk jumlah kesalahan data yang sedikit. Hal ini mensyaratkan pengkodean kanal untuk memproteksi data sebelum ditransmisikan pada kanal. Kode LDPC merupakan teknik pengkodean kanal yang mampu mendekati batas kapasitas *Shannon* (W. Xiang, 2003).

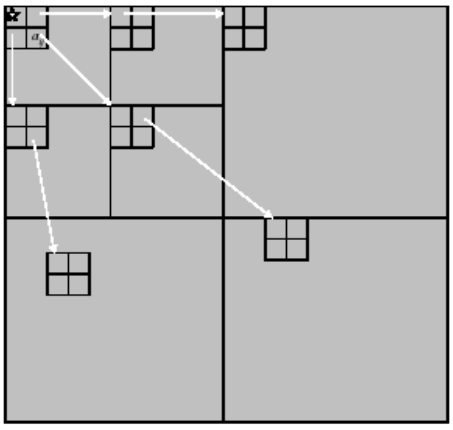
Pada artikel ini kami mengusulkan sebuah metode untuk memperbaiki performansi sistem transmisi citra pada kanal *noise*. Pada sisi pengirim digunakan teknik pengkodean gabungan SPIHT-LDPC yang mengalokasikan *bit budget* secara optimal menggunakan teknik UEP. Data yang penting diproteksi menggunakan kode LDPC dengan *code rate* yang lebih rendah. Dengan demikian pada data yang penting jarang terjadi *error*, sehingga citra dapat direkonstruksi dengan kualitas yang lebih baik.

Paper diorganisasi sebagai berikut: Bagian 2 menjelaskan secara singkat tentang algoritma SPIHT dan kode LDPC. Bagian 3 membahas mengenai metode yang diusulkan. Bagian 4 membahas hasil simulasi. Serta kesimpulan pada bagian 5.

2. KOMPRESI CITRA DAN PENGKODEAN KANAL

2.1 Algoritma Kompresi SPIHT

Algoritma SPIHT diusulkan oleh *Pearlman* dan *Said*, didasarkan pada transformasi *wavelet* (A. Said and W. A. Pearlman, 1996). Algoritma SPIHT secara intensif menggunakan struktur data dinamis dari koeffisien *wavelet*, untuk mengeksploitasi *self-similarities*. Hubungan *parent-child* pada koeffisien *wavelet* ditunjukkan pada Gambar 1.



Gambar 1. Ilustrasi hubungan *parent-child* dari koeffisien SPIHT.

Pada Algorima SPIHT koeffisien-koeffisien diklasifikasikan kedalam tiga set, yaitu:

- LIP (*list of insignificant pixel*) merupakan koordinat dari koeffisien yang tidak signifikan berdasarkan threshold saat ini.

- LSP (*list of significant pixel*) merupakan koordinat dari koeffisien yang tidak signifikan berdasarkan threshold saat ini.
- LIS (*list of insignificant sets*) merupakan koordinat dari akar dengan subpohon yang tidak signifikan.

Selama proses kompresi, set dari koeffisien pada LIS diperbaharui dan jika koeffisien menjadi signifikan dipindahkan dari LIP ke LSP. Dengan demikian *bitstream* dapat diorganisasi secara progressif.

Dengan cara yang sama set secara berurutan dievaluasi sesuai LIS, dan saat set yang ditemukan signifikan ia dihilangkan dari daftar dan dipartisi. Subset baru dengan lebih dari satu elemen ditambahkan kembali ke LIS, dengan set koordinat tunggal ditambahkan ke akhir LIP atau LSP, tergantung apakah mereka sigfikan atau tidak.

Algoritma pengkodean dinyatakan sebagai berikut:

1. ***Inisialisasi***: Keluaran n = $[\log_2(\max_{(i,j)}\{|c_{i,j}|\})]$, set LSP dengan isi kosong, dan tambahkan koordinat $(i,j) \in$ H ke LIP, dan jika turunannya juga ke LIS dengan tipe A.
2. ***Sorting pass***:
   2.1. Untuk setiap entri (i,j) pada LIP:
   2.1.1. Keluarkan $S_n(i,j)$;
   2.1.2. jika $S_n(i,j) = 1$, pindahkan (i,j) ke LSP dan keluarkan tanda dari $c_{i,j}$.
   2.2. Untuk setiap entry (i,j) pada LIS:
   2.2.1. Jika entri adalah tipe A:
   - Keluarkan $S_n(D(i,j))$.
   - Jika $S_n(D(i,j)) = 1$, maka:

   1. Untuk setiap $(k,l) \in O(i,j)$:
   - Keluarkan $S_n(k,l)$.
   - Jika $S_n(k,l) = 1$, tambahkan (k,l) ke LSP dan keluarkan tanda dari $c_{k,l}$.
   - Jika $S_n(k,l) = 0$, tambahkan (k,l) ke akhir dari LIP.

   2. Jika $L(I,j) \neq \emptyset$, pindahkan (i,j) ke akhir dari LIS, dan bila entry tipe B, menuju ke langkah 2.2.1, jika tidak hilangkan entry (i,j) dari LIS.

   2.2.2. Jika entri adalah tipe B:
   - Keluarkan $S_n(L(i,j))$.
   - Jika $S_n(L(i,j)) = 1$, maka:

   a) Tambahkan setiap $(k,l) \in O(i,j)$ ke akhir dari LIS jika entri tipe A.
   b) Hilangkan (i,j) dari LIS.

3. ***Refinement Pass***: untuk setiap entry (i,j) dalam LSP, kecuali yang termasuk pada sorting pass terakhir (pada n yang sama), keluarkan bit msb ke-n dari $|c_{i,j}|$.
4. ***Update*** langkah kuantisasi: kurangi n dengan 1, lalu kembali ke langkah 2.

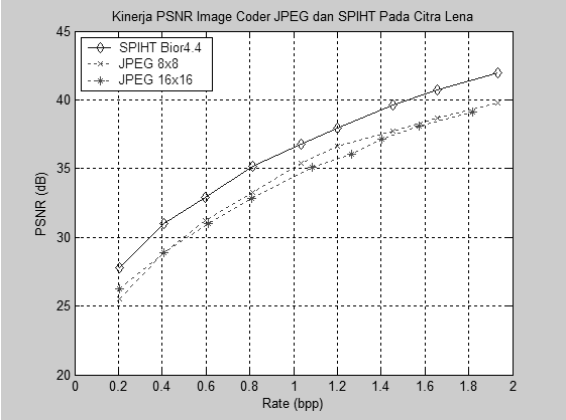
Algoritma pendekodean SPIHT menggunakan metode yang sama pengkodeannya, sehingga citra dapat direkonstruksi. Karena pada proses kompresi terjadi proses pemfilteran, maka citra hasil rekonstruksi akan mengalami distorsi. Distorsi dinyatakan dengan *mean square error* (MSE):

$$MSE=\left(\frac{1}{M.N}\right)\sum_{y=1}^{M}\sum_{x=1}^{N}[I(x,y)-I'(x,y)]^2 \qquad (1)$$

Kualitas citra hasil rekonstruksi dibandingkan dengan citra asal menggunakan parameter *peak signal to noise ratio* (PSNR). PSNR dihitung menggunakan persaman berikut:

$$PSNR = 20\log_{10}\left(\frac{2^{res}-1}{\sqrt{MSE}}\right) \qquad (2)$$

Semakin besar nilai PSNR, makan kualitas citra hasil rekonstruksi semakin baik. Hasil simulasi perbandingan kinerja algoritma kompresi SPIHT dan kompresi standar (JPEG) ditunjukkan pada Gambar 2. Dari Gambar 2 dapat ditunjukkan bahwa kompresi SPIHT memiliki nilai PSNR yang lebih tinggi dibandingkan dengan JPEG pada laju kompresi yang sama. Sehigga kinerja algoritma kompresi SPIHT lebih baik dari JPEG.



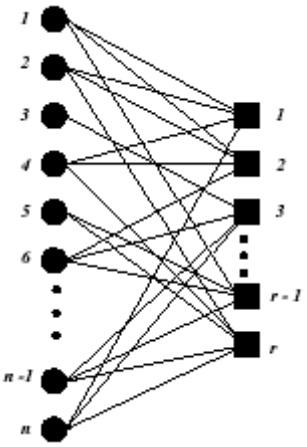
Gambar 2. Grafik perbandingan unjuk kerja algoritma kompresi SPIHT dengan JPEG.

2.2 Kode LDPC

Kode LDPC ditemukan oleh *Robert Gallager* dalam tesis PhDnya (R.G. Gallager, 1963) dan ditemukan ulang 30 tahun sesudahnya (I. M. Tanner, 1981), (S. Y. Chung, G. D. Forney, Jr. T. J. Richardson and R. Urbanke, 2001). Kode LDCP merupakan kode blok linier yang diperoleh dari *sparse bipartite graph* (*Tanner graph*). Kode LDPC menggunakan matriks cek paritas yang *sparse* (jumlah elemen *non-zero* yang sedikit). Graph terdiri dari *n message* atau *bit nodes* dan *r check nodes*. *Graph* memunculkan kode blok linier dengan panjang *n*. *Codeword* merupakan vektor $(c_1,c_2,...,c_n)$ yang mana untuk seluruh *check node*, jumlah posisi bersebelahan berdasarkan *message node* adalah nol. Ilustrasi contoh dari *Tanner Graph* ditunjukkan pada Gambar 3.

Tanner graph dari kode LDPC disebut *regular* jika setiap *message node* dihubungkan dengan

jumlah yang sama ke setiap *check node* dan setiap *check node* dihubungkan dengan jumlah yang sama ke *message node*. Jika tidak kode LDPC disebut dengan *irregular*.



Gambar 3. *Tanner Graph* dari kode LDPC.

Serupa dengan kode *Turbo*, kode LDPC menggunakan pendekodean iteratif memberikan performansi BER yang sangat baik dengan kompleksitas rendah (T. J. Richardson, and T. L. Urbanke, 2001). Kode LDPC *irregular* diketahui memiliki performansi yang lebih baik dari kode *Turbo* dan kode LDPC *regular*. Keuntungan dari kode LDPC *irregular* dibanding kode *Turbo* adalah proses dekoding lebih cepat, dan eror yang terjadi dapat dideteksi (J. Richardson, M. A. Shokrollahi and R. U. Urbanke, 2001).

Kode LDPC yang digunakan untuk simulasi adalah algoritma *sum-product*. Tujuan dari pendekodean *sum-product* adalah menghitung *a posteriori probability* (APP) untuk setiap bit *codeword* $P_i = P\{c_i = 1|N\}$, yang merupakan probabilitas bit *codeword* ke-*i* adalah 1 kondisional pada kejadian N dan seluruh persyaratan cek paritas terpenuhi. Probabilitas intrinsic (probabilitas a priori) P_i^{int}, merupakan probabilitas bit awal yang tidak tergantung pada persyaratan kode, dan probabilitas extrinsic P_i^{ext} didapat dari kejadian *N*.

Algoritma *sum-product* memiliki empat langkah sebagai berikut:

1. ***Inisialisasi***. Pesan inisial dikirim dari message node i ke check node j berupa LLR dari sinyal y_i (soft) yang diterima dari kanal. Untuk kanal AWGN dengan ***signal-to-noise ratio*** E_b/N_0:

$$L_{i,j} = R_i = 4y_i \frac{E_b}{N_0} \qquad (3)$$

2. ***Check-to-bit***: LLR extrinsic dari ***check node*** ke-***j*** untuk ***message node*** ke-i merupakan probabilitas cek paritas ke-j terpenuhi jika bit ke-***i*** diasumsikan 1.

$$E_{i,j} = \log_e\left(\frac{1+\prod_{i'\in B_{j,i'\neq i}} \tanh\left(L_{i',j}/2\right)}{1-\prod_{i'\in B_{j,i'\neq i}} \tanh\left(L_{i',j}/2\right)}\right) \qquad (4)$$

3. ***Codeword test***: Kombinasikan LLR dengan menjumlahkan LLR ekstrinsik dan LLR awal pada langkah 1.

$$L_i = \sum_{j\in A_i} E_{i,j} + R_i \qquad (5)$$

Pada setiap bit lakukan ***hard decision***:

$$z_i = \begin{cases} 1, & L_i \le 0 \\ 0, & L_i > 0 \end{cases} \qquad (6)$$

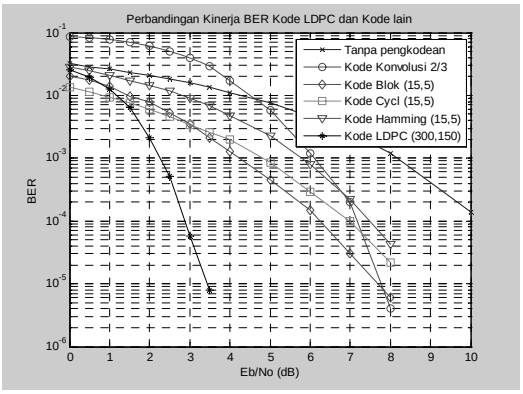
Jika $z = [z_1,\ldots,z_n]$ merupakan ***codeword*** yang valid ($H.z^T = 0$), atau jika jumlah maksimum iterasi yang dijinkan terlewati, algoritma dihentikan.

4. ***Bit-to-check***: Pesan dikirim dari setiap ***message node ke check node*** yang terhubung, kecuali ***message node*** ke-***i*** mengirim ke ***check node j***, menghitung LLR tanpa menggunakan informasi dari ***check node*** ke-***j***:

$$L_{i,j} = \sum_{j'\in A_{i,j'\neq j}} E_{i,j'} + R_i \qquad (7)$$

Kembali ke langkah 2.

Kinerja kode LDPC dibandingkan dengan pengkodean kanal lainnya ditunjukkan pada Gambar 4.



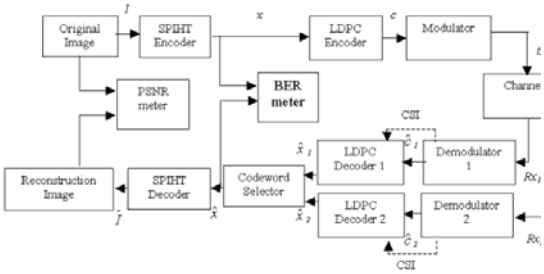
Gambar 4. Grafik perbandingan unjuk kerja kode LDPC dengan pengkodean kanal lainnya.

3. JOINT LDPC DECODING

Pada joint LDPC decoding (pendekodan gabungan LDPC), model kanal yang digunakan terdiri dari dua kanal paralel yang independen. Pada kanal ditransmisikan blok data yang sama, sehingga kedua pendekode (LDPC *decoder* 1 dan LDPC *decoder* 2) memproses data yang sama pada satu waktu proses pendekodean.

Diasumsikan bahwa probabilitas kedua kanal mengalami *noise* yang besar secara bersamaan adalah kecil. Sehingga bila salah satu pendekode LDPC gagal mendekodekan data, maka kita masih dapat memperoleh data dari pendekode LDPC

lainnya. Skema pendekode LDPC gabungan ditunjukkan pada Gambar 5.



Gambar 5. Diagram sistem pengkodean gabungan SPIHT-LDPC

Data keluaran dari *codeword selector* dipilih berdasarkan aturan:

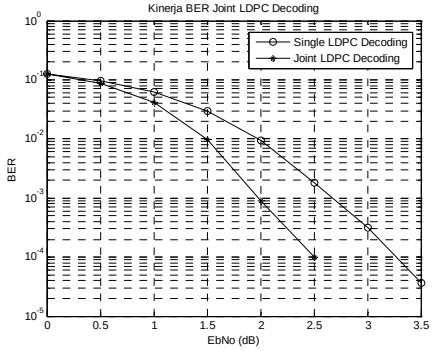
$$\hat{x} = \begin{cases} \hat{x}_1, & \hat{c}_1.H^T = 0 \\ \hat{x}_2, & \hat{c}_1.H^T \neq 0 \end{cases}$$

Data $\hat{x}$ kemudian didekompresi menggunakan algoritma dekompresi SPIHT sehingga diperoleh citra rekonstruksi ($\hat{I}$). Kinerja BER sistem pendekodean gabungan adalah dengan membandingkan data *x* dan data $\hat{x}$, sedangkan kinerja PSNR dihitung dengan membandingkan nilai *I* dan $\hat{I}$.

4. HASIL SIMULASI

Pada simulasi transmisi citra menggunakan teknik *joint* LDPC *decoding* (pendekodean LDPC gabungan), digunakan teknik EEP melalui dua kanal paralel independen. Citra dikompresi menggunakan algoritma SPIHT dengan laju kompresi 1,5 dan kode LDPC (300,150) dengan *code rate* 1/3. Kinerja BER dan PSNR sistem *joint* LDPC *decoding* disimulasikan menggunakan sistem yang ditunjukkan pada Gambar 6.

Dari Gambar 6, untuk mencapai BER 10^{-4} *joint* LDPC *decoding* membutuhkan Eb/No 2,5 dB, sedangkan untuk *single* LDPC *decoding* membutuhkan Eb/No 3,8 dB, maka *coding gain* dari penggunaan teknik *joint* LDPC *decoding* adalah 1,3 dB.



Gambar 6. Kinerja BER joint LDPC decoding

5. KESIMPULAN

Dari hasil simulasi, dapat diambil kesimpulan bahwa teknik joint LDPC decoding pada transmisi citra dapat meningkatkan kualitas citra rekonstruksi. Teknik ini mampu memberikan coding gain sebesar 3,8 dB untuk BER 10^{-4}. Penggunaan teknik *joint* LDPC *decoding* secara rata-rata memberikan peningkatan PSNR sebesar 4,44 dB. Penelitian ini selanjutnya akan dikembangkan menggunakan pengkodean sumber multi deskripsi, serta penggunaan teknik modulasi *orthogonal frequency division multiplexing* (OFDM) pada kanal frekuensi selektif.

PUSTAKA

W. Xiang,"Joint source-channel coding for image transmission and related topic", *PhD Thesis*, Institute for Telecommunications Research Univer-sity of South Australia, Dec. 2003.

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 Tech.*, vol. 6, pp. 243-250, June 1996.

R. G. Gallager, *Low-Density Parity-Check Codes*, Cambridge, MA: M.I.T. Press, 1963.

I. M. Tanner, "Recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 533-547, Sep. 1981.

S. Y. Chung, G. D. Forney, Jr., T. J. Richardson and R. Urbanke, "On the design of low-density parity check codes within 0.0045 dB of Shannon limit," *IEEE Comm. Lett.*, vol. 5, no. 2, pp. 58-60, Feb. 2001.

T. J. Richardson and T. L. Urbanke, "The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding," *IEEE Trans. Inform. Theory*, vol. 2, Feb. 2001.

J. Richardson, M. A. Shokrollahi and R. U. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 619 – 637, Feb. 2001.