UNIVERSITAS INDONESIA

KOMPRESI VIDEO MEDIS BERDASARKAN ALGORITMA H.264 DENGAN TRANSFORMASI WAVELET DISKRIT

SKRIPSI

ANNISA DINDA AMALIA
0806315824

FAKULTAS TEKNIK UNIVERSITAS INDONESIA
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
JULI 2012

UNIVERSITAS INDONESIA

KOMPRESI VIDEO MEDIS BERDASARKAN ALGORITMA H.264 DENGAN TRANSFORMASI WAVELET DISKRIT

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana

ANNISA DINDA AMALIA
0806315824

FAKULTAS TEKNIK UNIVERSITAS INDONESIA
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
JULI 2012

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama : Annisa Dinda Amalia

NPM : 0806315824

Tanda Tangan :

Tanggal : 13 JUNI 2012

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:
Nama	: Annisa Dinda Amalia
NPM	: 0806315824
Program Studi	: Teknik Elektro
Judul Skripsi	: Kompresi Video Medis Berdasarkan Algoritma H.264 Dengan Transformasi Wavelet Diskrit

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro, Fakultas Teknik, Universitas Indonesia

DEWAN PENGUJI

Pembimbing : Prof. Dr. Ir. Dadang Gunawan, M.Eng ( )

Penguji : Dr. M. Suryanegara, S.T., M.Sc. ( )

Penguji : Dr. Ir. Arman Djohan Diponegoro ( )

Ditetapkan di : Depok
Tanggal : 21 Juni 2012

KATA PENGANTAR

Puji dan syukur penulis panjatkan ke hadirat Allah SWT karena hanya dengan limpahan berkat dan rahmat-Nya penulis dapat menyelesaikan skripsi ini. Tak lupa shalawat dan salam kepada junjungan Nabi Muhammad SAW yang telah mengantarkan cahaya terang kepada umatnya hingga akhir zaman. Penulis menyadari bahwa tanpa dukungan dari berbagai pihak, akan sulit bagi penulis untuk bisa menyelesaikan skripsi ini dengan baik. Oleh karena itu, penulis mengucapkan terima kasih kepada:

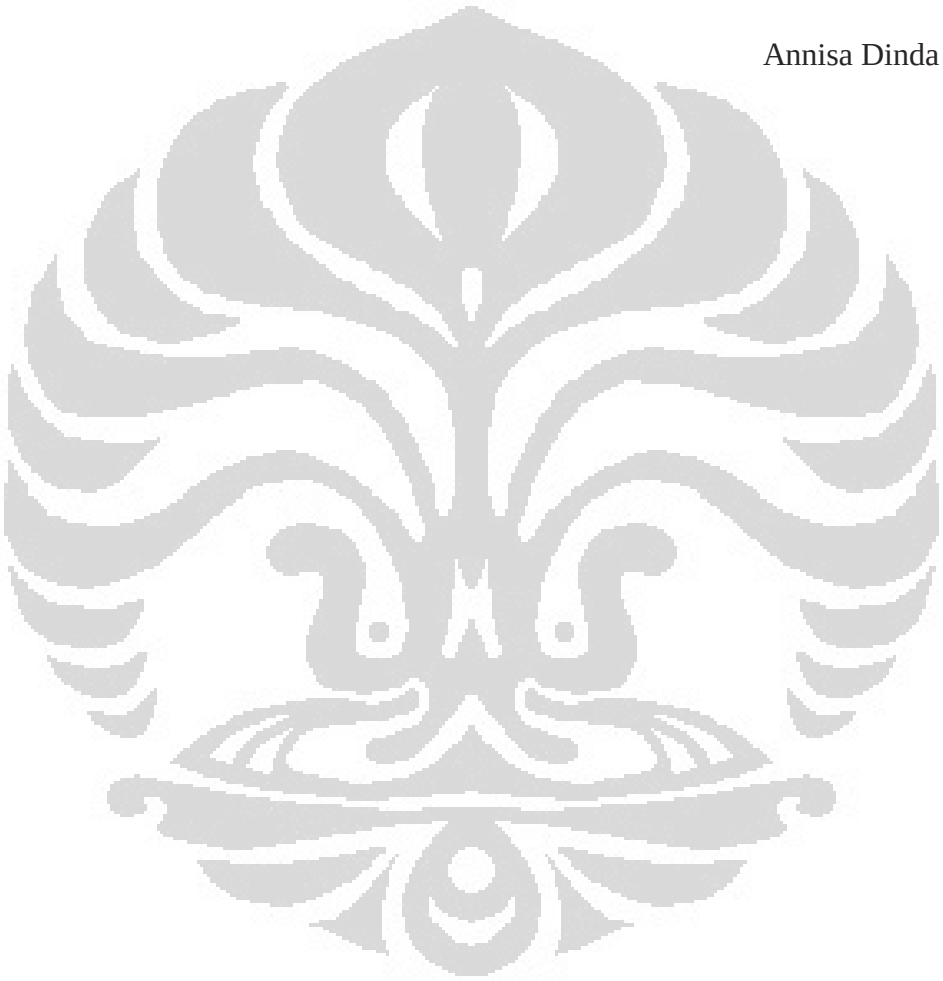
1. Prof. Dr. Ir. Dadang Gunawan, M. Eng. sebagai pembimbing yang dengan sabar banyak memberikan arahan dan wawasan sehingga penulis dapat menyelesaikan skripsi ini dengan baik;
2. Filbert H. Juwono, S.T, M.T sebagai dosen yang sering mengingatkan dan menanyakan perkembangan skripsi ini;
3. Ayah, Mama, dan adik-adik tercinta yang selalu memberikan dukungan moral dan spiritual, menghibur saat penulis sedang jenuh, bahkan seringkali menemani penulis menyelesaikan skripsi;
4. rekan-rekan Asisten Laboratorium Telekomunikasi DTE FTUI, khususnya angkatan 2008 dan 2009 : Pipit, Renita, Syifa, Idham, Mayendra, Nugroho, Budin;
5. rekan-rekan satu bimbingan : Chandra, Randy, Arif, Sofwan, dan bapak-bapak mahasiswa S3;
6. rekan-rekan Elkom 08, khususnya peminatan telekomunikasi, Elkom08 yang aktif di jejaring sosial Twitter dan Facebook, Fast Track DTE FTUI 2008, Cewek Elkom 08 : Renita, Syifa, Pipit, Iput, Lya, Wuri, Apil, Rani, Hana, Wenni, Fira, Megu, Ai, Aneta, Vela, Gardin, Ipul, Noni, Dyani, Rika, Sarrah, Ninis;
7. rekan-rekan S2 Teknik Telekomunikasi;
8. sepupu-sepupuku, Laras, Dini, Desandri, yang rajin menyemangati via SMS;
9. sahabat-sahabatku, Tia, Enci, Fando, Ijal, dkk, yang selalu mendoakan dan memberi dukungan bagi penulis dalam menyelesaikan skripsi;

10. warga DTE serta seluruh pihak yang telah membantu terselesaikannya skripsi ini yang tidak dapat penulis sebutkan satu persatu.

Akhir kata, penulis berharap agar Allah SWT berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat.

Depok, 13 Juni 2012

Annisa Dinda Amalia



HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI
TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini :

Nama : Annisa Dinda Amalia
NPM : 0806315824
Program Studi : Teknik Elektro
Departemen : Teknik Elektro
Fakultas : Teknik
Jenis Karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif** (*Non-exclusive Royalty Free Right*) atas karya ilmiah saya yang berjudul :

Kompresi Video Medis Berdasarkan Algoritma H.264
Dengan Transformasi Wavelet Diskrit

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (database), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.
Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 13 JUNI 2012
Yang menyatakan

(Annisa Dinda Amalia)

ABSTRAK

Nama : Annisa Dinda Amalia
Program Studi : Teknik Elektro
Judul : Kompresi Video Medis Berdasarkan Algoritma H.264 Dengan Transformasi Wavelet Diskrit

Kebutuhan transmisi data *telemedicine* seperti citra dan video medis secara nirkabel mengharuskan data dikompresi terlebih dahulu agar dapat dikirimkan secara efisien. Yu, et.al, membuktikan bahwa standar kompresi H.264 dapat diterapkan pada kompresi video medis, sementara menurut Xiong, et.al transformasi wavelet diskrit mengungguli DCT pada kompresi citra dan video. Skripsi ini bertujuan untuk mensimulasikan algoritma H.264 dengan transformasi 2D berupa transformasi wavelet diskrit serta menganalisis hasil simulasinya dengan parameter kualitas objektif yaitu PSNR dan rasio kompresi. Simulasi kompresi video medis berdasarkan algoritma H.264 dengan DCT maupun DWT dilakukan pada parameter kompresi yang sama untuk dibandingkan hasilnya. Hasil simulasi menunjukkan bahwa DWT dapat diterapkan pada kompresi video medis berdasarkan algoritma H.264 dengan *gain* hingga 1,06 dB mengungguli DCT. Namun demikia, rasio kompresi yang didapat tidak terlalu berbeda secara signifikan akibat adanya *padding* pada hasil dekomposisi DWT.

Kata kunci :
Kompresi video, H.264, transformasi kosinus diskrit, transformasi wavelet diskrit, video medis

ABSTRACT

Name : Annisa Dinda Amalia
Study Program : Electrical Engineering
Title : Medical Video Compression Based-on H.264 Algorithm Using Discrete Wavelet Transform

The need of telemedicine data transmission, such as image and video, using wireless network requires data to be compressed efficiently. Yu, et.al, proved that H.264 standard of video coding is applicable in medical video compression, while according to Xiong et.al, discrete wavelet transform outperforms discrete cosine transform in image and video compression. This thesis aims to simulate medical video compression based-on H.264 algorithm using discrete wavelet transform and analyze the simulation results showed by objective quality parameter, namely PSNR and compression ratio. The simulation using DCT and DWT performed on the same compression parameters to compare the results. The simulation result show that DWT is applicable in medical video compression based-on H.264 algorithm with 1,06 dB outperformed DCT. However, the compression ratio obtained doesn't differ significantly because of the pixel padding on the DWT decomposition matrices.

Key words:
Video compression, H.264, discrete cosine transform, discrete wavelet transform, medical video

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Kemajuan sistem komunikasi nirkabel saat ini, baik di negara maju maupun negara berkembang seperti Indonesia memberikan kesempatan bagi bidang lain untuk mengembangkan teknologinya dan mengaplikasikannya melalui komunikasi nirkabel. Terlihat dengan teknologi LTE maupun WiMAX yang mampu memberikan kecepatan *transfer* data tinggi hingga mencapai 100 Mbps. Dengan kecepatan *transfer* data yang begitu tinggi, memungkinkan layanan multimedia menjadi dominasi dalam komunikasi nirkabel saat ini. Keunggulan ini juga dimanfaatkan untuk meningkatkan pelayanan kesehatan di daerah-daerah yang jauh dari sumber daya ahli dan fasilitas yang lengkap.

Topik *telemedicine* selaras dengan peningkatan pelayanan kesehatan bagi masyarakat yang berada di daerah yang terpencil dimana salah satu kendalanya adalah kurangnya dokter spesialis. *Telemedicine* sendiri didefinisikan sebagai penggunaan telekomunikasi untuk menyediakan informasi medis maupun layanan medis [1]. Saat ini bidang yang paling banyak diaplikasikan dalam *telemedicine* ialah *teleradiology*, yaitu pengiriman gambar sinar X, *CT scan*, atau *MRI*, bahkan video hasil *ultrasound*.

Seperti diketahui bahwa dokter spesialislah yang lebih berkompeten dalam menganalisis serta mendiagnosis kondisi pasien berdasarkan video medis dan data medis 3-D (seperti *CT, MRI, echocardiography, angiography*, video hasil *ultrasound* dan lain sebagainya). Dengan demikian, *telemedicine* dapat menjadi salah satu solusi dalam rangka peningkatan pelayanan kesehatan bagi masyarakat yang berada di daerah yang terpencil yang tidak terjangkau oleh dokter spesialis.

Berdasarkan data tahun 2002, jumlah dan distribusi dokter spesialis di Indonesia sangat tidak merata. Statistik data menyatakan bahwa jumlah dokter spesialis di Indonesia baru berjumlah 9.205 orang. Sebagian besar dokter spesialis

bekerja di Jakarta, yakni sebanyak 2.441 orang yang merupakan 27% dari jumlah dokter spesialis di Indonesia [2]. Kecenderungan ini menyebabkan minimnya pelayanan kesehatan yang memadai di daerah-daerah terpencil. Padahal, setiap warga negara Indonesia memiliki hak yang sama dalam mendapatkan pelayanan kesehatan.

Dengan semakin berkembangnya teknologi transmisi yang diterapkan, maka diharapkan transmisi data medis pasien tersebut dapat cepat diterima oleh ahlinya tanpa kurang sesuatu apapun untuk mencegah kesalahan diagnosis. Namun bagaimanapun, implementasi teknologi komunikasi pita lebar seperti LTE dan WiMAX tidak menjamin data medis yang asli dari peralatan yang digunakan dapat dikirim dengan cepat. Hal ini disebabkan oleh ukuran data medis yang begitu besar mencapai ratusan Mbps bahkan Gbps, sehingga melebihi kapasitas *bandwidth* yang disediakan. Sebagai contoh, video hasil *ultrasound* gerakan *fetus* yang berdurasi 30 sekon memiliki ukuran *file* hingga 10 MB [1].

Untuk mengatasi hal tersebut data medis yang ditransmisikan haruslah melewati tahapan kompresi. Kompresi data medis yang demikian itu, haruslah dilakukan dengan metode yang tepat. Berbeda dengan kompresi video biasa dimana kualitas masih bisa ditoleransi dengan rasio kompresi, kompresi video medis membutuhkan kualitas tinggi serta rasio kompresi yang tinggi [3]. Sebab untuk keperluan diagnosis, sangatlah penting bahwa proses kompresi tidak menyebabkan hilangnya detail penting pada citra dan memunculkan artifak yang dapat menyebabkan kesalahan interpretasi serta diagnosis. Di sisi lain, batasan kapasitas *bandwidth* yang disediakan juga harus diperhatikan agar data video medis tersebut dapat dikirimkan dengan cepat.

Berbagai metode kompresi telah dikembangkan untuk mengatasi *trade-off* yang terjadi dalam kompresi video medis seperti halnya dalam kompresi citra medis. Sejak tahun 1994, Tsai et al. telah mengembangkan skema kompresi untuk video *angiogram* berdasarkan *full frame discrete wavelet transform* [4]. Kemudian Gibson et al. mengajukan skema kompresi menggunakan transformasi wavelet secara *lossy* untuk data video *angiogram* [5]. Ada pula model kompresi *lossless* pada ROI (*Region of Interest*) atau bagian yang dianggap penting untuk diagnosis pada citra dalam video. Namun, kelemahan dari metode-metode diatas

adalah tidak adanya standar yang berlaku seperti perangkat lunak yang disediakan sehingga sulit untuk diterapkan. DICOM, *Digital Imaging and Communications in Medicine* yang memberi fasilitas standar format *file* tertentu untuk bertukar data pencitraan medis secara *universal*, pun akhirnya merekomendasikan JPEG serta MPEG-4. Namun, seiring dengan berkembangnya standar kompresi video yang kini sudah diperbaiki dengan kemunculan H.264, maka standar ini pun patut untuk diuji coba pada video medis. Sebab sejak kemunculannya, H.264 sangat memadai untuk banyak aplikasi seperti komunikasi video dua arah, pemancar televisi, *video streaming* melalui IP, penyimpanan media digital, dan lain sebagainya.

Pengujian kompresi video medis dengan standar kompresi H.264 telah dilakukan sejak tahun 2005 oleh Yu et.al [3]. Yu melakukan uji coba penerapan standar H.264 untuk kompresi video medis berupa *CT* dan *echocardiography*. Berdasarkan eksperimen yang dilakukannya, H.264 terbukti melakukan kompresi lebih baik daripada MPEG-4 yang pernah diuji coba oleh Lau et.al. pada tahun 2000 untuk data *ultrasound*. Dalam paper tersebut dikatakan bahwa diperoleh hasil *gain* PSNR yang meningkat dan terjadi penurunan *bit rate*.

Keunggulan H.264 memberikan pencerahan dalam kompresi video medis. Namun demikian, standar kompresi tersebut masih menggunakan transformasi kosinus diskrit (DCT). Transformasi tersebut memiliki kelemahan karena dapat memunculkan artifak akibat adanya efek *blocking* pada proses transformasi. Disamping itu, berbagai penelitian pun telah menunjukkan adanya transformasi lain yang dapat mengatasi kelemahan transformasi kosinus diskrit. Xiong et.al. [6] pada 1999 pernah membandingkan performa antara DCT dengan transformasi wavelet pada kompresi citra dan video. Berdasarkan eksperimen tersebut, terlihat bahwa transformasi wavelet mengungguli DCT dengan *gain* PSNR sebesar 0,5 dB pada video dan 1,5 dB pada citra biasa.

Referensi [7][8] dan [9] juga menunjukan bahwa aplikasi transformasi wavelet pada aplikasi kompresi citra dan video maupun aplikasi pada kompresi citra medis lebih baik dibandingkan dengan transformasi DCT dalam hal rasio kompresi dan rekonstruksi gambar.

1.2 Perumusan Masalah

Uji coba terhadap kompresi video medis dengan H.264 sudah dilakukan dan hasilnya menunjukkan keunggulan dibandingkan dengan standar yang digunakan DICOM yaitu MPEG-4. Namun demikian, transformasi wavelet memiliki keunggulan dibandingkan dengan DCT yang masih digunakan dalam H.264. Oleh karena itu, dirancanglah teknik kompresi video medis dengan menerapkan standar kompresi H.264 dengan menggunakan transformasi wavelet diskrit.

Masalah yang akan dibahas pada skripsi ini adalah:

a. Bagaimana teknik penerapan algoritma kompresi H.264 berdasarkan transformasi wavelet diskrit pada video medis?
b. Bagaimana nilai PSNR hasil kompresi berdasarkan algoritma H.264 dengan transformasi wavelet diskrit?
c. Bagaimana nilai rasio kompresi hasil kompresi berdasarkan algoritma H.264 dengan transformasi wavelet diskrit?

1.3 Tujuan

Tujuan skripsi ini adalah untuk mensimulasikan dan menganalisis kompresi video medis menggunakan algoritma H.264 berdasarkan transformasi wavelet diskrit.

1.4 Batasan Masalah

Permasalahan yang akan dibahas pada skripsi ini dibatasi pada penelitian kompresi video medis berdasarkan algoritma kompresi H.264 dengan transformasi wavelet diskrit.

Batasan-batasan permasalahan yang akan dibahas pada skripsi ini antara lain ialah:

1. Data video diambil dari website www.cipr.rpi.edu/ dalam bentuk urutan citra medis berformat RAS.
2. Simulasi dibuat mengikuti algoritma H.264 untuk prediksi spasial dan temporal hingga tahap kuantisasi; sedangkan *deblocking filter* dan *entropy coding* tidak diikutsertakan untuk mengurangi kompleksitas.

3. Kualitas performansi dari simulasi yang dibuat tidak diuji dengan kualitas subjektif.
4. Nilai PSNR dan rasio kompresi tidak digeneralisasi untuk setiap tipe video.

1.5 Sistematika Penulisan

Pembahasan dalam skripsi ini adalah:

BAB 1 Pendahuluan

Bab 1 berisi gambaran umum permasalahan yang diangkat pada penelitian. Bab 1 terdiri dari latar belakang, perumusan masalah, tujuan penelitian, batasan masalah, dan sistematika penulisan skripsi.

BAB 2 Kompresi Video Menggunakan H.264

Bab 2 berisi dasar teori mengenai kompresi video menggunakan standar kompresi H.264.

BAB 3 Perancangan Simulasi

Bab 3 berisi perancangan simulasi teknik kompresi video medis dengan menerapkan standar kompresi H.264 berdasarkan transformasi wavelet diskrit.

BAB 4 Hasil Simulasi dan Analisis

Bab 4 berisi data hasil simulasi serta analisis hasil simulasi dengan penilaian kualitas objektif berupa parameter PSNR dan rasio kompresi dari data urutan citra MRI dan CT.

Bab 5 Kesimpulan

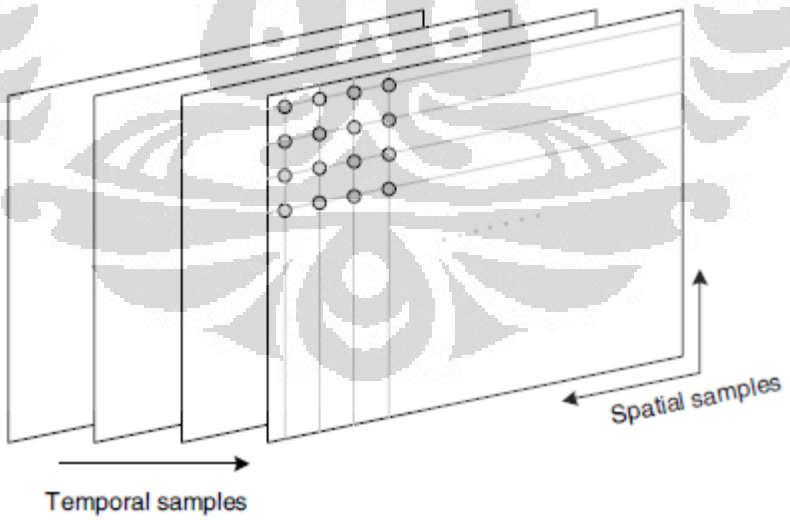
Bab 5 berisi kesimpulan dari skripsi yang diajukan.

BAB 2
KOMPRESI VIDEO MEDIS MENGGUNAKAN H.264

2.1 Video

Video pada dasarnya adalah urutan dari banyak citra digital yang ditampilkan pada suatu *frame rate* tertentu untuk menciptakan ilusi animasi. Citra-citra digital diperlihatkan sesuai urutan dengan jangka waktu tertentu sehingga tampak bergerak. Dalam video dikenal *frame* yang merupakan citra-citra digital. Oleh karena itu, pengolahan video juga melibatkan proses pengolahan citra.

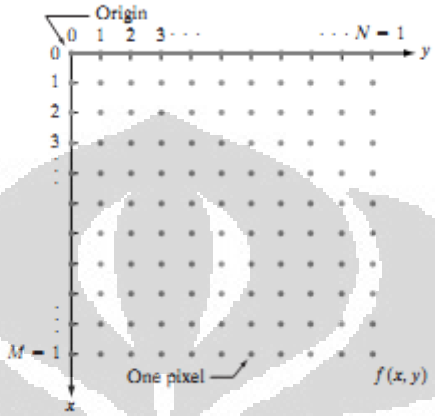
Video sering disebut sebagai citra tiga dimensi karena terdiri dari dua dimensi spasial yang mewakili citra atau *frame* video serta satu dimensi waktu atau temporal yang mewakili pergerakan *frame* seperti yang terlihat pada Gambar 2.1. Oleh karena itu, pengkodean dalam video pun menjadi dua yakni pengkodean spasial dan temporal.



Gambar 2. 1 Definisi video [10]

Domain spasial dari video diwakili oleh citra digital yang dapat dimodelkan sebagai matriks berukuran $M \times N$, dengan M dan N adalah ukuran

baris dan kolom pada matriks, serta $M \times N$ merupakan resolusi dari citra. Setiap koordinat dari $M \times N$ disebut sebagai piksel atau pel (*picture element*), dimana setiap piksel memiliki nilai yang merepresentasikan intensitas citra [11]. Representasi domain spasial pada suatu *frame* dapat dilihat pada Gambar 2.2.



Gambar 2. 2 Domain spasial dalam suatu *frame* [11]

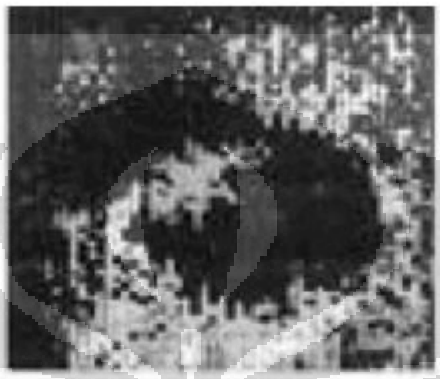
2.2 Video Medis

Selain citra medis, salah satu jenis data medis lainnya yang sering digunakan untuk diagnosis kesehatan pasien oleh seorang dokter ialah dengan video medis. Video medis ini sebenarnya sama saja dengan pencitraan medis, namun ada banyak citra yang diambil per sekon kemudian kumpulan citra tersebut ditampilkan sehingga dapat dilihat perubahan yang terjadi pada objek diagnosis. Video medis yang dimaksud dalam dunia kesehatan ialah hasil rekam *ultrasound*, misalnya *echocardiography*. Citra medis 3-D juga turut dikelompokkan ke dalam video medis di antaranya adalah CT dan MRI, misal *angiography* [3]. Keunggulan dari diagnosis dengan menggunakan video medis tentunya ialah kapabilitas dokter untuk menganalisis dengan lebih baik karena dapat diperhatikan pula perubahan yang terjadi pada objek diagnosis. Namun ukuran data yang dimiliki lebih besar pula daripada citra medis.

2.2.1 *Ultrasound*

Ultrasound ialah suatu teknik pencitraan medis dengan menggunakan gelombang audio dengan frekuensi di atas frekuensi pendengaran manusia.

Penggunaan *ultrasound* untuk pencitraan medis dilakukan oleh Karl Dussek bersama saudaranya Friedrich Dussek pada 1938 yang menghasilkan suatu *hyperphonogram* otak seperti pada Gambar 2.3 [12]. Selanjutnya pada 1957 S. Satomura [12] mengembangkan metode Doppler untuk mengukur kecepatan aliran darah.



Gambar 2. 3 Hyperphonogram otak yang diambil pada 1938 [12]

Teknik ini dilakukan dengan meletakkan suatu transduser elektromekanis berupa piezoelektrik pada permukaan tubuh yang memancarkan pulsa-pulsa *ultrasound* dengan frekuensi tengah tertentu yang merambat menuju jaringan tubuh. Sinyal pantulan dari jaringan tubuh kemudian direkam oleh transduser tersebut untuk mensintesis citra dari bagian tubuh yang diinginkan [12]. Gelombang yang digunakan ialah gelombang suara dengan frekuensi di atas frekuensi pendengaran manusia, biasanya antara 2,5-10 MHz [13].

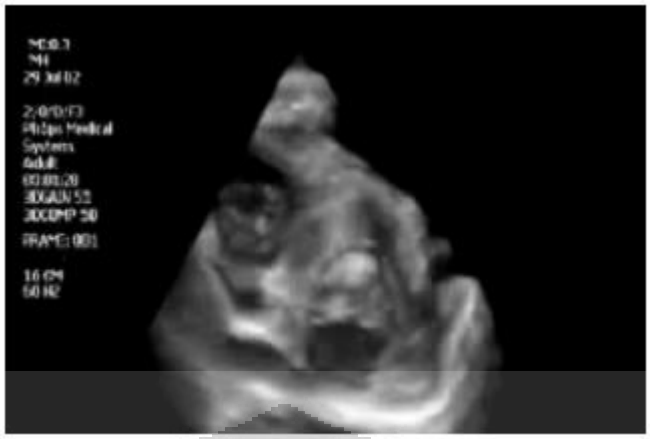
Gambar 2. 4 Hasil USG janin 10 minggu [12]

Kelebihan dari *ultrasound* dibanding dengan CT dan MRI adalah hasil pencitraanya yang *real-time*, harganya murah, dan ukurannya yang kecil dan ringan sehingga mudah dibawa-bawa. Kekurangannya terletak pada kualitas citra yang terbatas [12]. Jenis *ultrasound* paling umum digunakan ialah dalam dunia kebidanan yakni *obstetric ultrasound* yang digunakan untuk mengamati janin dalam kandungan seperti pada Gambar 2.4. Pengembangan *ultrasound* 3D dilakukan dengan menggunakan *phase array tranducer* dua dimensi sehingga dapat diaplikasikan pada diagnosis penyakit pembuluh darah.

2.2.2 *Echocardiography*

Echocardiography atau dikenal juga dengan *cardiac ultrasound* ialah suatu teknik *ultrasound* untuk mengamati struktur dan kondisi jantung. Pertama kali dilakukan oleh Inge Edler terhadap Hellmuth Hertz dari *Lund University* pada 1953 yang mendemonstrasikan pencatatan terus menerus gerakan dinding jantung dengan alat sonar [14]. Kemudian dibuat dalam bentuk film yang disebut sebagai *echocardiograph* pada 1960. Teknologi terbaru *echocardiography* menerapkan sistem 3D *real-time imaging* sehingga hasilnya dapat disimpan dalam bentuk video medis serta dapat dilakukan manipulasi. Gambar 2.5 menunjukkan suatu hasil USG dengan menggunakan 3D *real-time imaging*.

Echocardiography 3D bisa dilakukan dari luar (*trans thorakal echocardiography* / TTE) atau dari dalam melalui tenggorokan (*trans esophageal echocardiography* / TEE). Kualitas gambar TEE lebih baik dari TTE. Namun biasanya untuk meningkatkan kejernihan hasil dari TTE diinjeksikan *contrast agent* ke aliran darah.



Gambar 2. 5 Hasil USG Jantung dalam bentuk 3D [12]

2.2.3 *Angiography*

Angiography atau *arteriography* ialah teknik pencitraan medis yang digunakan untuk mengamati bagian pembuluh darah arteri. Pertama kali dikembangkan oleh neurolog Egas Moniz pada 1927 yang melakukan *angiography* pada otak dengan menggunakan sinar X [15].

Secara tradisional, *angiography* dilakukan dengan menginjeksikan suatu jenis cairan sebagai *contrast-agent* pada bagian pembuluh darah dengan menggunakan kateter kemudian disinari dengan sinar X. Bagian yang diinjeksikan cairan tersebut akan menyerap sinar X sehingga menghasilkan warna gelap. Selain dengan sinar X kini *angiography* dapat dilakukan dengan menggunakan MRI. Untuk seluruh struktur kecuali jantung, pencitraan dilakukan dengan metode *Digital Subtraction Angiography* (DSA) dimana citra yang diambil sebanyak 2-3 *frame* per sekon sehingga dapat dievaluasi aliran darah dalam pembuluh tanpa terganggu oleh gambar tulang dan organ yang tidak dibutuhkan lainnya. Untuk jantung, pencitraan dilakukan sebanyak 15-30 *frame* per sekon namun bukan dengan teknik DSA [15].

2.2.4 MRI

MRI atau *Medical Resonance Imaging* ialah suatu teknik pencitraan organ dalam tubuh yang menggunakan medan magnet untuk mengarahkan atom hidrogen dalam tubuh [12]. Pasien akan ditempatkan di dalam suatu magnet besar sehingga dapat mengarahkan atom hidrogen yang terdapat dalam tubuh pasien

kemudian dengan frekuensi radio mengubah arah magnetisasi tersebut sehingga perubahan arahnya dapat dikenali oleh *scanner* dan direkonstruksi menjadi citra dari area yang dideteksi [12]. MRI lebih aman daripada pencitraan dengan sinar X atau pun CT.

Dibandingkan dengan CT, MRI menghasilkan kontras lebih baik pada jaringan lunak serta tidak adanya radiasi ion. Terlebih lagi medan magnet yang kuat dan konstan tidak memiliki efek negatif bagi tubuh. Kekurangannya adalah teknologi dan instalasi yang membutuhkan biaya tinggi, serta terbatas pada pasien dengan kondisi tertentu [12].

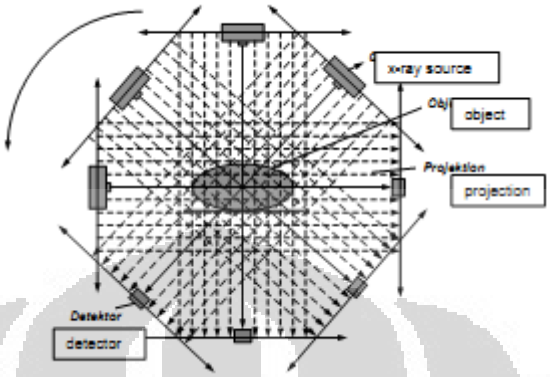
Saat ini MRI sudah diaplikasikan pada banyak bidang, antara lain untuk *angiography* dan *obstetric* seperti yang terlihat pada Gambar 2.6. *Real-time* MRI dilakukan untuk memonitor secara kontinyu objek yang bergerak misalkan jantung dan persendian. Teknik *real-time* MRI ialah berdasarkan radial *FLASH* dan *iterative reconstruction* yang menghasilkan resolusi temporal antara 20-30 ms untuk citra dengan resolusi 1.5-2 mm.

Gambar 2. 6 *Angiography* dengan MRI [12]

2.2.5 CT

CT atau *Computed Tomography* pertama kali diperkenalkan oleh G.N. Hounsfield pada 1973 [12]. Pencitraan CT menggunakan sinar X dengan energi foton berkisar antara 30-200 KeV. Metode CT terdiri dari pengambilan sekumpulan proyeksi sinar X pada bagian tubuh yang dituju dengan berbagai posisi sudut seperti pada Gambar 2.7. Pasien ditempatkan di dalam suatu tabung yang berisi pemancar sinar X dan detektor untuk menangkap sinar X yang melewati tubuh. Tabung tersebut dapat berputar sehingga proyeksi yang ditangkap

detektor mengambil gambar bagian tubuh yang berbeda-beda. Kemudian data dari detektor diolah sehingga dihasilkan citra bagian tubuh yang lebih jelas untuk dianalisis [12]. Gambar 2.8 menunjukkan contoh citra hasil CT scan.



Gambar 2. 7 Prinsip CT *scanner* generasi pertama [12]

Gambar 2. 8 Citra hasil CT

2.3 *Discrete Cosine Transform*

Discrete Cosine Transform (DCT) adalah teknik mengubah suatu sinyal dalam domain waktu menjadi komponen frekuensi dasar [16]. Biasanya DCT digunakan untuk kompresi citra. Dikembangkan oleh Ahmed, Natarajan, dan Rao pada 1974, DCT dapat dikatakan sebagai bagian *real* dari *Discrete Fourier Transform* (DFT). Pertama kali digunakan untuk kompresi citra oleh Chen dan Pratt pada 1984 [16]. Pada kompresi citra maupun video, data residu hasil *motion-compensation* akan ditransformasi ke dalam domain lain. DCT merupakan salah satu transformasi yang banyak digunakan dalam algoritma kompresi citra maupun video.

DCT dilakukan pada suatu matriks sampel X berukuran NxN dan menghasilkan matriks koefisien Y berukuran NxN. *Forward* DCT maupun *inverse* DCT dapat dilakukan dengan menggunakan operasi matematika terhadap matriks DCT yaitu matriks A dengan matriks blok X sehingga menghasilkan matriks koefisien Y. Untuk *forward* DCT, persamaan matematisnya ialah sebagai berikut [10]:

$$Y = AXA^T \tag{2-1}$$

Sedangkan untuk *inverse* DCT, persamaan matematisnya ialah sebagai berikut [10]:

$$X = A^T YA \tag{2-2}$$

Elemen dari matriks A adalah[10] :

$$A_{ij} = C_i \cos\frac{(2j+1)i\pi}{2N} \tag{2-3}$$

dimana $C_i = \sqrt{\frac{1}{N}} \quad (i = 0) \quad ; \qquad C_i = \sqrt{\frac{2}{N}} \quad (i > 0)$

Sehingga persamaan matematis matriks di atas dapat disederhanakan menjadi [10]:

$$Y_{xy} = C_x C_y \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} X_{ij} \cos\frac{(2j+1)y\pi}{2N} \cos\frac{(2i+1)x\pi}{2N} \tag{2-4}$$

$$X_{ij} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} C_x C_y Y_{xy} \cos\frac{(2j+1)y\pi}{2N} \cos\frac{(2i+1)x\pi}{2N} \tag{2-5}$$

Gambar 2. 9 Citra dengan ukuran 8x8 piksel [16]

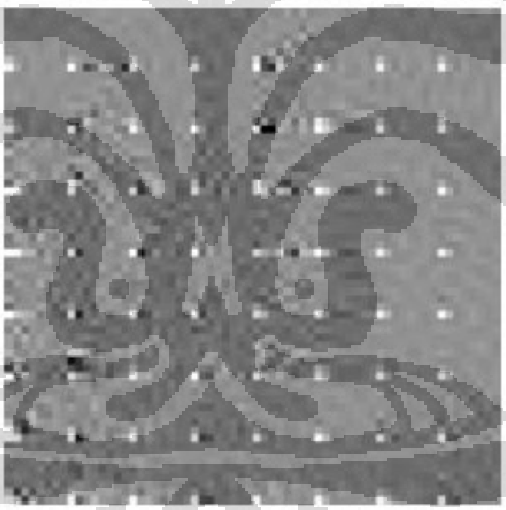
Gambar 2. 10 Hasil DCT dari Gambar 2.9 [16]

Gambar 2.9 merupakan contoh suatu citra yang akan ditransformasi dengan DCT dan Gambar 2.10 ialah hasil DCT 2D dari Gambar 2.8. Piksel pada hasil DCT menunjukkan proporsi dari masing-masing fungsi basis 2D yang mewakili citra *input*. Piksel yang memiliki warna paling terang dikenal sebagai DC frekuensi {0,0}. Frekuensi DC memiliki nilai piksel rata-rata dari seluruh nilai piksel pada citra *input*, dan biasanya merupakan koefisien terbesar dari hasil DCT.

Biasanya DCT dilakukan pada citra yang telah dibagi ke dalam blok-blok berukuran tertentu. Dengan melakukan transformasi pada setiap blok secara paralel dibutuhkan memori untuk perhitungan yang lebih sedikit sehingga proses kompresi lebih sederhana dan cepat. DCT yang dilakukan dengan *blocking* ini dikenal dengan sebutan *blocked DCT* [16].

Contohnya ialah citra pada Gambar 2.11 yang berukuran 64x64 piksel diproses dengan *blocked* DCT berukuran 8x8 piksel.

Gambar 2. 11 Citra dengan ukuran 64x64 piksel yang akan diproses dengan *blocked DCT* [16]



Gambar 2. 12 Hasil *blocked* DCT Gambar 2.11 [16]

Dapat dilihat pada Gambar 2.12, terdapat titik-titik terang yang menunjukkan koefisien DC dari masing-masing *block* DCT.

Karena menggunakan fungsi kosinus, maka matriks hasil transformasi DCT hanya bergantung pada koefisien frekuensi rendahnya saja atau pada matriks ialah elemen-elemen awal dari matriks tersebut, karena koefisien lainnya akan bernilai sangat kecil atau nol. Prinsip inilah yang digunakan untuk kompresi, yaitu merepresentasikan citra hanya dengan beberapa koefisien saja. Keunggulan dari

DCT ialah tingkat kompleksitas perhitungan yang rendah sehingga cepat untuk komputasi serta mudah dalam implementasi. Namun pada *blocked* DCT, seringkali muncul artifak akibat efek *blocking* akibat proses transformasi yang dilakukan dalam ukuran blok tertentu. Gambar 2.13 menunjukkan citra yang memiliki artifak akibat *blocked* DCT.

Gambar 2. 13 Citra yang memiliki artifak akibat *blocked* DCT [10]

2.4 Transformasi Wavelet

Transformasi wavelet mulai digunakan dalam aplikasi pengolahan dan analisis sinyal pada pertengahan 1980 untuk menginvestigasi sinyal seismik oleh Pierre L. Goupillaud [17]. Berbeda dengan transformasi Fourier yang menggunakan fungsi basis berupa sinusoidal, transformasi wavelet menggunakan basis wavelet (*small waves*). Dengan menggunakan transformasi wavelet, suatu sinyal dapat dianalisis dalam domain frekuensi dan waktu sekaligus.

Ide dasar dari transformasi wavelet adalah merepresentasikan fungsi f sebagai jumlah produk dari fungsi-fungsi yang disebut wavelet. Setiap wavelet diperoleh dari suatu fungsi wavelet induk dengan melakukan *scaling* dan translasi [18].

2.4.1 Transformasi Wavelet Kontinyu

Wavelet ialah suatu gelombang kecil yang energinya terkonsentrasi dalam suatu durasi tertentu. Wavelet memiliki karakteristik osilasi seperti gelombang biasa namun hanya memiliki durasi tertentu, tidak beraturan, dan asimetrik [19].

Transformasi wavelet kontinu dilakukan dengan menggunakan *window* yang dapat di*scaling* dan ditranslasikan [17]. *Window* tersebut disebut sebagai wavelet induk. Wavelet induk adalah suatu fungsi yang memenuhi kriteria matematika sebagai berikut [17]:

(1) Harus memiliki energi yang terbatas

$$E = \int_{-\infty}^{\infty} |\psi(t)|^2 \, dt \; < \infty \tag{2-6}$$

(2) Jika $\hat{\psi}(f)$ adalah transformasi Fourier dari $\psi(t)$

$$\hat{\psi}(f) = \int_{-\infty}^{\infty} \psi(t) e^{-j(2\pi f)t} \, dt \tag{2-7}$$

maka harus memenuhi persamaan berikut

$$C_g = \int_0^{\infty} \frac{|\hat{\psi}(f)|^2}{f} df \; < \infty \tag{2-8}$$

Persamaan (2-8) dikenal sebagai kondisi *admissibility* dan nilai C_g disebut sebagai konstanta *admissibility*. Nilai tersebut tergantung dari wavelet induk yang dipilih.

(3) Untuk wavelet kompleks, maka transformasi Fouriernya harus *real* dan tidak memiliki frekuensi negatif.

Secara matematis transformasi wavelet dari suatu sinyal kontinu $x(t)$ didefinisikan sebagai [17]:

$$T(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t) \psi^*\left(\frac{t-b}{a}\right) dt \tag{2-9}$$

dimana a adalah konstanta *scaling* dan b adalah konstanta translasi.

2.4.2 Transformasi Wavelet Diskrit

Secara umum, transformasi wavelet diskrit menggunakan *dyadic grid* dari nilai *a* dan *b* serta basis fungsi wavelet orthonormal dan menghasilkan *zero-redundancy* [17]. Pada transformasi wavelet diskrit, maka nilai *a* dan *b* harus disampel dengan menggunakan pendiskritan logaritmik basis 2 sehingga di dapatkan *dyadic grid* dengan persamaan berikut [17]

$$\psi_{m,n}(t) = 2^{-m/2} \, \psi(2^{-m} t - n) \tag{2-10}$$

dimana *m* dan *n* mengatur *scaling* dan translasi dari wavelet tersebut.

Persamaan matematis dari transformasi wavelet diskrit ialah sebagai berikut [17]

$$T_{m,n} = \int_{-\infty}^{\infty} x(t)\psi_{m,n}(t)dt \quad (2\text{-}11)$$

dengan $T_{m,n}$ adalah koefisien wavelet pada skala dan lokasi m, n.

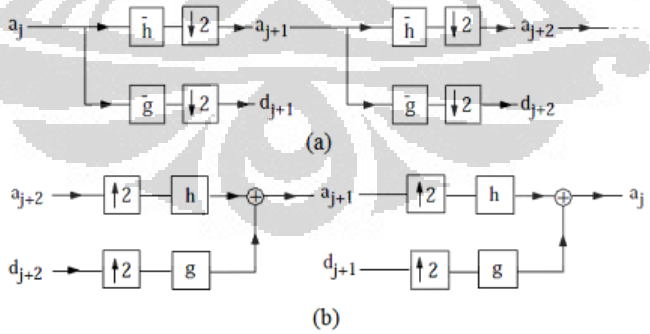
2.4.3 *Filter Bank*

Transformasi wavelet yang banyak digunakan dalam kompresi citra ialah berdasarkan jumlah koefisien filter yang ekuivalen dengan fungsi wavelet diskrit atau disebut dengan *filter bank*. Sepasang filter diterapkan pada sinyal sehingga terbagi menjadi *band* frekuensi rendah (L) dan *band* frekuensi tinggi (H). Setiap *band* ter-subsampel dengan faktor dua sehingga masing-masing *band* frekuensi terdiri dari N/2 sampel.

Filter bank umum digunakan untuk sinyal dua dimensi misalkan untuk intensitas citra. Setiap baris pada citra dua dimensi difilter dengan filter *low-pass* dan *high-pass* (Lx dan Hx) dan *output* dari masing-masing filter ter-subsampel dengan faktor dua sehingga menghasilkan citra *intermediate* L dan H. L ialah citra asli yang difilter *low-pass* dan ter-subsampel pada arah x, sementara H ialah citra asli yang difilter *high-pass* dan ter-subsampel pada arah x. Selanjutnya setiap kolom dari citra baru ini difilter dengan cara yang sama (Ly dan Hy) serta ter-subsampel dengan faktor dua sehingga menghasilkan empat buat sub-citra (LL, LH, HL, HH). 'LL' menunjukkan citra asli, merupakan hasil filter *lowpass* pada arah horizontal dan vertikal. 'HL' ialah hasil filter *high-pass* pada arah vertikal dan mengandung residu frekuensi vertikal. 'LH' ialah hasil filter *high-pass* pada arah horizontal dan mengandung residu frekuensi horizontal. Sementara 'HH' ialah hasil filter *high-pass* pada arah vertikal dan horizontal. Seluruh *band* frekuensi tersebut mengandung informasi mengenai citra asli, namun *band* LH, HL, dan HH menunjukkan sifat *sparse* sehingga dapat digunakan dalam prinsip kompresi. Gambar 2.14 merupakan contoh citra hasil dekomposisi dengan transformasi wavelet diskrit.

Gambar 2. 14 Citra hasil dekomposisi dengan transformasi wavelet diskrit [10]

Pada aplikasi kompresi citra, biasanya dilakukan dekomposisi wavelet berulang-ulang pada *subband* LL sehingga membentuk pohon *subband*. *Subband* frekuensi yang lebih tinggi dari hasil pohon *subband* biasanya bernilai mendekati nol sehingga dapat dihilangkan dalam proses kompresi. Penggunaan *filter bank* dapat dilihat pada Gambar 2.15. Keuntungan dari penggunaan *filter bank* ialah lebih mudah dalam implementasi pada perangkat keras.



Gambar 2. 15 Contoh skema *filter bank* [20]

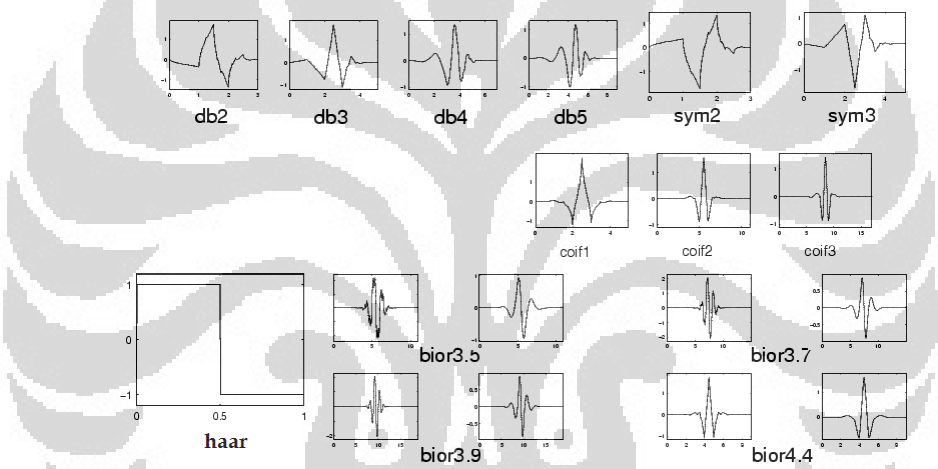
2.4.4 Jenis Wavelet Induk

Wavelet induk memiliki bentuk dan karakteristik bermacam-macam. Berbagai jenis wavelet induk diteliti untuk mendapatkan karakteristik yang

mampu mengakomodir kebutuhan pada pemrosesan sinyal digital yaitu agar dapat menghasilkan rekonstruksi yang sempurna.

Ada berbagai jenis wavelet induk yang sering digunakan dalam pengolahan sinyal digital terutama citra. Di antaranya ialah Haar, Daubechies, Coiflet, Symlet, dan Biorthogonal. Jenis-jenis wavelet induk ini banyak digunakan karena memiliki *fast algorithm* untuk menyederhanakan komputasi serta dibangun dari suatu FIR filter yang mudah untuk diimplementasikan.

Masing-masing jenis wavelet induk yang telah disebutkan sebelumnya, memiliki karakteristik orthogonalitas, simetrik, dan *compactness* yang berbeda-beda. Gambar 2.16 menunjukkan bentuk wavelet induk yang berbeda.



Gambar 2. 16 Berbagai jenis wavelet induk [21]

2.5 Kompresi Video

Kompresi adalah suatu teknik untuk memperkecil ukuran data dengan mengurangi bit-bit yang redundan sehingga dapat disimpan dan ditransmisikan secara efisien. Beberapa teknik yang digunakan untuk mereduksi data pada suatu video antara lain dengan mengurangi nuansa warna pada citra, mengurangi resolusi warna, menghilangkan bagian kecil atau detail yang tidak perlu, dan membandingkan citra yang berurutan kemudian menghilangkan detail yang tidak berubah antara dua citra tersebut [22].

Ada dua jenis kompresi yaitu *lossless* dan *lossy*. Kompresi *lossless* akan menghasilkan data hasil kompresi sama persis dengan data asli sebab tidak ada bit redundan yang dihilangkan. Namun kompresi *lossless* tidak bisa menghasilkan

rasio kompresi yang besar. Sedangkan kompresi *lossy* akan menghasilkan data hasil kompresi yang terlihat sama dengan data asli namun tidak sama persis dengan data asli (terdapat artifak), sebab beberapa bit yang redundan dihilangkan.

Kompresi video terdiri dari dua prinsip dasar. Pertama ialah berdasarkan redundansi spasial yang terdapat dalam setiap *frame*. Kedua ialah karena suatu *frame* video kebanyakan mirip dengan *frame* sebelum atau sesudahnya sehingga dapat dikatakan memiliki redundansi temporal. Kompresi video diawali dengan mengkodekan *frame* pertama seperti melakukan kompresi citra biasa. Kemudian untuk *frame* selanjutnya dapat dilakukan dengan mengidentifikasi perbedaan dengan *frame* sebelumnya. Perbedaan inilah yang dikodekan sebagai pengganti representasi *frame* tersebut. Namun jika terdapat banyak perbedaan, maka lebih baik jika *frame* tersebut dikodekan seperti melakukan kompresi citra biasa. Pengkodean perbedaan antar *frame* disebut dengan teknik *interframe coding*, sedangkan pengkodean *frame* dengan prinsip kompresi citra disebut dengan teknik *intraframe coding*.

Intraframe coding dilakukan dengan melakukan transformasi 2D pada *frame* kemudian hasil transformasi tersebut dikuantisasi. Setelah didapat matriks baru, maka koefisien-koefisien yang tidak bernilai nol diurutkan dengan metode *zig-zag scanning*. Setelah didapat urutan koefisien maka selanjutnya dilakukan *entropy coding*. *Entropy coding* yang dilakukan biasanya dengan RLE atau Huffman. Pada *intraframe coding* juga dapat dilakukan suatu *intra-prediction* dimana nilai piksel dari suatu citra diprediksi dari nilai piksel yang bersebelahan dalam citra tersebut.

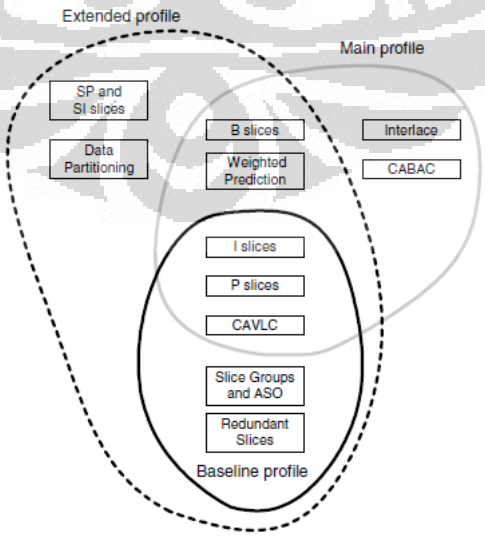
Interframe coding dilakukan dengan mencari *motion vector* dari *frame* baru terhadap *frame* referensi. Metode yang dilakukan ialah dengan membandingkan blok-blok pada kedua *frame* tersebut sehingga didapatkan arah perubahan posisi blok tersebut. Setelah itu, jika selain perubahan posisi terdapat pula perubahan sudut atau penskalaan, maka dilakukan *error compensation*. Hal ini dilakukan dengan membandingkan *frame* referensi yang diberi *motion vector* dengan *frame* aslinya sehingga didapatkan *error* atau selisih keduanya. Dalam *interframe coding*, koefisien yang dikodekan dengan *entropy coding* hanyalah

motion vector dan *error compensation*nya saja. Dengan demikian, hanya butuh sedikit koefisien untuk merepresentasikan dua buah *frame*.

2.6 Standar Kompresi H.264

ITU-T H.264 / MPEG-4 (Part 10) *Advanced Video Coding* merupakan seri terbaru dari standar pengkodean video internasional. H.264/AVC dikembangkan oleh *Joint Video Team* (JVT) yang terdiri dari ahli dari ITU-T *Video Coding Experts Group* (VCEG) dan ISO/IEC *Moving Picture Experts Group* (MPEG). Standar pengkodean video ini didesain untuk memiliki keseimbangan antara efisiensi pengkodean, kompleksitas implementasi, serta biaya yang dibutuhkan. Bahkan keunggulan dari standar pengkodean video dari segi efisiensi pengkodean mencapai 2-3 kali dari MPEG-2 yang banyak diterapkan hingga saat ini [23].

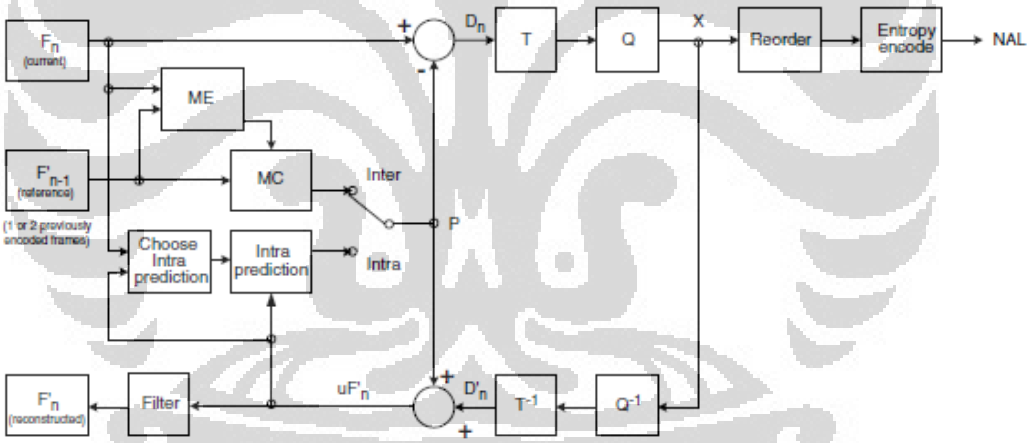
H.264 memiliki tiga *profile*, seperti yang ditunjukkan oleh Gambar 2.17, yang terdiri dari fungsi *encoder* yang berbeda-beda serta digunakan untuk keperluan yang berbeda. *Baseline Profile* merupakan *profile* paling sederhana dari standar H.264 yang mendukung intra dan inter-*coding* (I dan P *slices*) serta pengkodean entropi CAVLC. Di samping itu, terdapat juga *Main Profile* dan *Extended Profile* yang mendukung performa kompresi lebih tinggi namun mempertahankan kualitas hasil rekonstruksi yang tinggi pula.



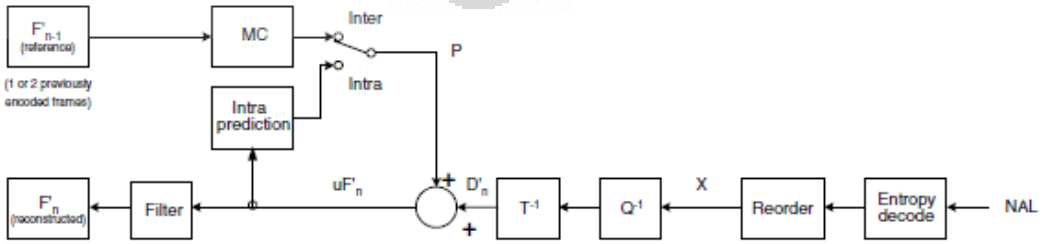
Gambar 2. 17 Profil yang terdapat dalam standar H.264 [10]

2.6.1 Skema Pengkodean

Secara umum, skema *encoder* dan *decoder* dari H.264/AVC tidak jauh berbeda dengan para pendahulunya. Skema *encode*r dan *decoder* H.264/AVC ditunjukkan pada Gambar 2.18 dan Gambar 2.19. H.264/AVC masih menggunakan transformasi DCT serta *motion-compensation*. Suatu *frame* dikompresi dengan membagi ke dalam beberapa *macroblock*. Setiap *macroblock* terdiri dari 16x16 sampel *luma* dengan sampel *chroma* menyesuaikan dengan rasio Y:Cb:Cr (4:2:0). Kemudian setiap *macroblock* dibagi lagi menjadi *sub-macroblock* untuk memudahkan dalam prediksi *motion-compensation*. Ukuran dari *sub-macroblock* ini memiliki tujuh variasi yakni 16x16, 16x8, 8x16, 8x8, 8x4, 4x8, dan 4x4. Sedangkan untuk transformasi spasial, blok yang digunakan berukuran 8x8 atau 4x4 piksel.



Gambar 2. 18 Arsitektur *encoder* H.264/AVC [10]



Gambar 2. 19 Arsitektur *decoder* H.264/AVC [10]

2.6.1.1 *Slice*

Pada H.264 setiap *frame* tidak harus dikodekan seluruhnya dalam satu jenis prediksi saja, spasial atau temporal. Setiap *frame* dapat dibagi lagi ke dalam *slice*. Setiap *slice* diprediksi berbeda-beda, bisa secara spasial atau temporal.

- *I-slice*

Pada H.264/AVC dikenal adanya *I-slice, P-slice, B-slice, SP-slice* atau *SI-slice*. Pada *I-slice* dilakukan prediksi spasial untuk nilai piksel yang berdekatan. Kemudian setelah melakukan prediksi spasial, informasi residu ditransformasi dengan blok berukuran 4x4 atau 8x8 lalu dikuantisasi. Kemudian koefisien hasil transformasi yang sudah dikuantisasi di*scan* dengan salah satu cara yakni *zig-zag* atau *field scan*. Setelah itu dikodekan dengan menggunakan CAVLC atau CABAC. Pada *I-slice* tidak terjadi prediksi temporal.

- *P-Slice*

Pada *P-slice* digunakan prediksi temporal dengan mengestimasi gerakan antara dua *frame*. Suatu gerakan dapat diestimasi dari *macroblock* 16x16 atau *sub-macroblock* yang lebih kecil. Estimasi gerakan dilakukan dengan menginterpolasi nilai piksel untuk mendapatkan tingkat akurasi ¼ piksel untuk *luma* dan 1/8 piksel untuk *chroma*. Interpolasi untuk *luma* dilakukan dengan memfilter piksel dengan *kernel* [1 -5 20 20 -5 1]/32 secara horizontal atau vertikal kemudian merata-ratakan dua nilai hasil pemfilteran yang bersebelahan.

Setelah melakukan prediksi temporal, dilakukan transformasi, kuantisasi, *scanning*, dan *entropy coding* seperti pada *I-slice*. Yang dikodekan hanyalah residu dari *frame* referensi dengan hasil prediksi saja.

- *B-slice*

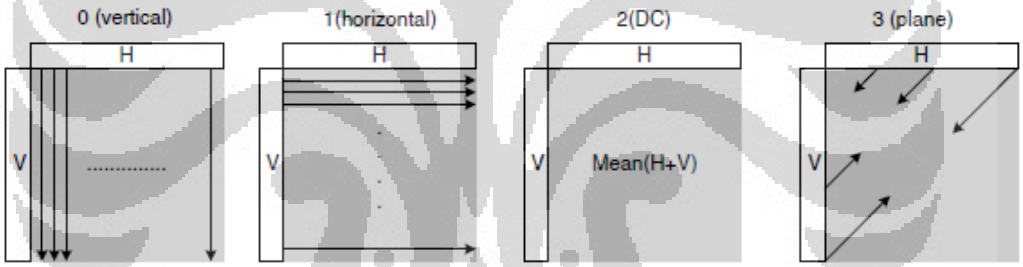
Pada *B-slice* terdapat dua *motion vector* yang merepresentasikan dua estimasi gerakan per *macroblock* untuk suatu prediksi temporal. Referensi *frame* dapat dari *frame* sebelum atau sesudahnya pada urutan tampilan video.

2.6.1.4 Prediksi Intra Spasial

Ada tiga tipe prediksi intra spasial pada H.264/AVC :

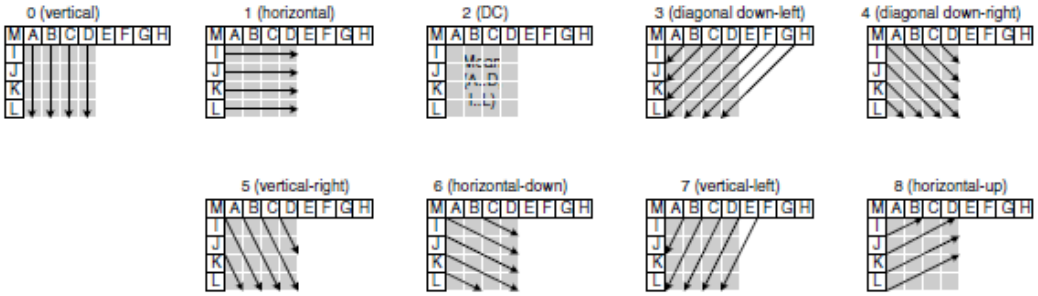
- Prediksi *full-macroblock* untuk 16x16 blok *luma* atau ukuran blok *chroma*
- Prediksi 8x8 *luma*
- Prediksi 4x4 *luma*

Pada prediksi *full-macroblock*, nilai piksel dari seluruh *macroblock luma* maupun *chroma* diprediksi dari piksel tepi yang berdekatan dengan *macroblock* lain. Prediksi ini dapat dilakukan dengan empat mode yaitu arah vertikal, horizontal, nilai rata-rata (DC), dan planar seperti pada Gambar 2.20. Untuk prediksi vertikal dan horizontal, maka *macroblock* diprediksi dari piksel yang ada di atas atau kiri dari *macroblock*. Sedangkan pada prediksi DC, maka nilai *luma* dari piksel yang berdekatan dirata-ratakan lalu nilai tersebut digunakan sebagai nilai prediksi. Pada prediksi planar, ada tiga parameter yang digunakan yaitu tingkat kecerahan, gradien arah horizontal dan vertikal yang diperkirakan cocok dengan piksel yang bersebelahan.



Gambar 2. 20 Mode prediksi nilai piksel pada prediksi intra *full-macroblock* 16x16 [10]

Prediksi 4x4 *luma* prinsipnya hampir mirip dengan prediksi *full-macroblock*. Perbedaannya adalah ukuran piksel yang diprediksi sebesar 4x4 dan prediksi dilakukan dengan sembilan mode seperti pada Gambar 2.21. Prediksi 8x8 memiliki metode yang sama dengan prediksi 4x4. Sedangkan untuk nilai chroma, maka prediksi yang dilakukan hanya dengan prediksi *full-macroblock* karena ukuran blok *chroma* kecil.



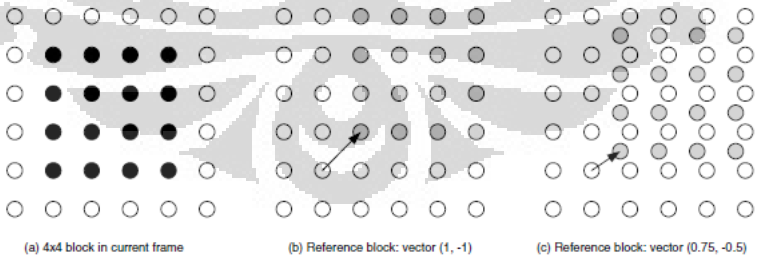
Gambar 2. 21 Mode prediksi spasial dengan blok luma 4x4 [10]

2.6.1.5 Prediksi Temporal

Prediksi temporal dilakukan hanya pada *slice* P, B, dan SP. Prediksi temporal pada H.264 sama seperti standar kompresi sebelumnya yaitu dengan *block-based motion compensation*. Setiap *macroblock* 16x16 dapat dibagi lagi ke dalam *sub-macroblock* berukuran 8x16, 16x8, 8x8, 4x8, 8x4, hingga 4x4. *Motion compensation* akan dilakukan pada setiap *sub-macroblock* sehingga dapat meyesuaikan ukuran bagian yang akan diprediksi secara temporal.

- *Motion Vector*

Motion vector pada H.264 memiliki akurasi hingga ¼ sampel. Maksudnya, pergerakan kecil dapat terprediksi dengan tepat karena nilai sampel diinterpolasi sebelumnya. Prediksi *motion vector* pada H.264 dapat dilihat pada Gambar 2.22.



Gambar 2. 22 Contoh prediksi *motion vector integer* dan sub-sampel [10]

Untuk mencari *motion vector* yang tepat dapat menggunakan beberapa algoritma pencocokan blok yang sudah banyak digunakan pada standar kompresi sebelumnya. Beberapa algoritma pencocokan blok tersebut antara lain ialah *Exhaustive Search, Three Step Search, New Three Step Search, Diamond Search,*

Adaptive Rood Pattern Search, dan lain sebagainya. Algoritma pencocokan yang paling dasar dan menghasilkan *motion vector* paling tepat ialah *Exhaustive Search* karena dilakukan pencarian *motion vector* di setiap piksel di daerah pencarian [24]. Namun algoritma tersebut memerlukan komputasi yang sangat banyak sehingga jarang digunakan.

- Prediksi *motion vector*

Jika prediksi temporal dilakukan dengan *sub-macroblock* yang kecil maka *motion vector* yang dihasilkan akan sangat banyak dan meningkatkan jumlah bit yang dibutuhkan. Dengan demikian, maka *motion vector* dapat pula diprediksi dari *motion vector* pada *sub-macroblock* yang bersebelahan karena kedua *motion vector* tersebut biasanya berkorelasi tinggi. Kemudian yang disimpan atau dikirim ialah perbedaan dari kedua *motion vector* tersebut.

2.6.1.6 Transformasi dan Kuantisasi

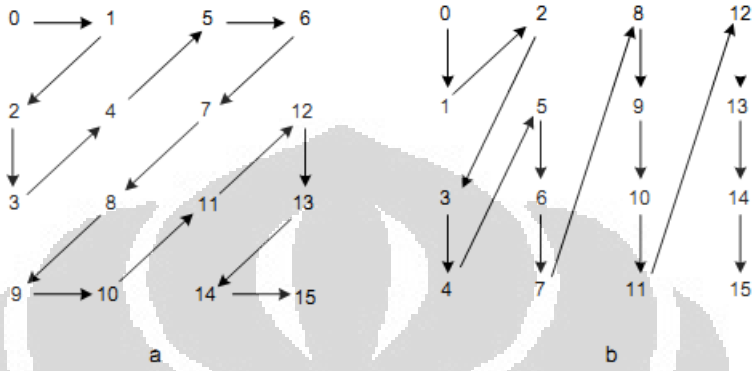
Setelah dilakukan prediksi spasial, dilakukan transformasi dari matriks yang merepresentasikan *frame* tersebut. Transformasi yang digunakan ialah dengan transformasi berdasarkan DCT terhadap masing-masing *macroblock* yang telah ditentukan. Hal ini dilakukan sehingga dapat mempercepat proses perhitungan koefisien hasil transformasi.

Setelah ditransformasi, koefisien-koefisien hasil transformasi dikuantisasi dengan menggunakan parameter kuantisasi yang dapat diubah untuk tiap *macroblock*. Parameter ini memiliki nilai antara 1-52 untuk format video yang memiliki 8 bit per sampel. Dengan parameter ini, maka dapat diatur bagian penting mana yang tidak boleh dikuantisasi terlalu besar.

2.6.1.7 *Scanning*

Koefisien hasil kuantisasi dapat di*scanning* dengan dua cara yaitu dengan *zig-zag scanning* maupun dengan *field scanning*. *Zig-zag scanning* diterapkan pada transformasi mode *frame* yaitu dengan mengurutkan koefisien dengan nilai terbesar terlebih dahulu dan koefisien yang tidak bernilai nol di awal urutan. Dengan *scanning* ini, maka sisa koefisien yang bernilai nol dapat berada di posisi

belakang dan dapat diabaikan atau tidak ikut dikodekan. Sedangkan *field scanning* pada prinsipnya sama saja, namun diterapkan jika transformasi dilakukan pada mode *field*. Yang umum digunakan ialah dengan menggunakan *zig-zag scanning*. Metode *zig-zag scanning* dan *field scanning* terdapat pada Gambar 2.23.



Gambar 2. 23 Metode *scanning* (a) *zig-zag scanning* (b) *field scanning* [23]

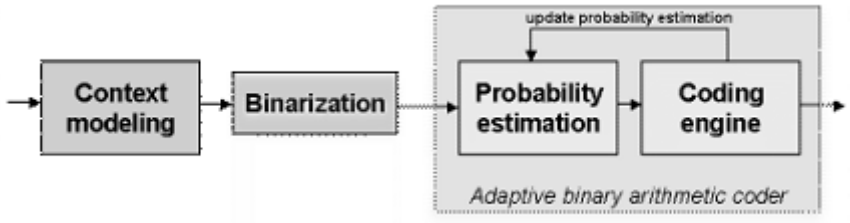
2.6.1.8 *Entropy coding*

Entropy coding adalah teknik pengkodean *lossless* yang menggantikan elemen data dengan representasi kode sehingga secara signifikan dapat mengurangi ukuran data. Jenis *entropy coding* yang digunakan ialah *variable length coding* (VLC) dan *binary arithmetic coding* (BAC). Pada H.264/AVC kedua *entropy coding* ini didesain untuk digunakan dalam teknik *context adaptive* (CA). *Context adaptive entropy coding* ini hanya digunakan untuk koefisien hasil transformasi, sedangkan untuk koefisien lain yang akan dikodekan menggunakan *entropy coding* yang tetap.

Entropy coding yang sering digunakan ialah VLC. Pada prinsipnya, VLC dilakukan ketika elemen data yang akan dikodekan (baik koefisien transformasi yang sudah dikuantisasi atau *motion vector*) muncul dalam frekuensi yang berbeda. Elemen data yang sering muncul dapat dikodekan dengan kode yang pendek, sedangkan yang jarang muncul dapat dikodekan dengan kode yang panjang. VLC yang sering digunakan antara lain adalah Huffman *code*.

Entropy coding CABAC memiliki performa lebih baik daripada CAVLC namun kompleksitas yang lebih tinggi serta harga yang lebih mahal. Efisiensi

kompresi dari *entropy coding* ini mencapai 10% lebih baik daripada CAVLC. Gambar 2.24 menunjukkan blok diagram *entropy coding* CABAC.



Gambar 2. 24 Blok diagram *encode*r CABAC [23]

2.6.1.9 *In-loop Deblocking Filter*

H.264 memiliki fitur *in-loop deblocking filter* yang bertujuan untuk mengurangi artifak akibat *blocking* dari prediksi *motion vector* maupun *blocked* DCT. *Filtering* dilakukan pada setiap macroblok yang sudah didekodekan atau sudah mengalami *rescaling* dan *inverse transform*. Dengan *filtering*, maka hasil rekonstruksi akan menjadi lebih baik.

Prinsip filtering pada H.264 terdiri dari tiga tahapan yaitu menentukan *boundary strength*, menentukan filter yang akan digunakan berdasarkan *boundary strength*, dan melakukan *filtering*. Penerapan kekuatan filter bersifat adaptif terhadap nilai *boundary strength* dari *sub-macroblock*. Kekuatan filter juga bergantung dari nilai parameter kuantisasi. Dengan demikian, kesalahan *filtering* seperti melakukan filter pada batas antara objek dan latar sehingga gambar menjadi *blur* dapat terhindar.

2.7 Kualitas Hasil Kompresi

Suatu hasil kompresi dapat diukur performansinya dengan beberapa parameter baik dari segi hasil rekonstruksi maupun seberapa besar kompresi yang terjadi. Hasil rekonstruksi kompresi dapat diukur baik secara visual (kualitas) maupun secara kuantitas. Pengukuran kualitas hasil rekonstruksi kompresi dapat dilakukan secara subjektif maupun objektif. Kualitas hasil rekonstruksi kompresi objektif yang baik belum tentu menghasilkan kualitas subjektif yang baik pula.

2.7.1 *Peak Signal-to-Noise Ratio* (PSNR)

PSNR merupakan salah satu parameter kualitas hasil rekonstruksi kompresi secara objektif. PSNR (dihitung dalam dB) merupakan nilai logaritmik dari nilai *Mean Square Error* (*MSE*) antara citra asli dan hasil rekonstruksi terhadap nilai maksimum yang mungkin pada citra. Apabila diketahui suatu matriks citra asli C berukuran $N \times N$ dan matriks citra rekonstruksi R dengan ukuran matriks $N \times N$, maka nilai PSNR dari citra hasil rekonstruksi dapat dihitung dengan rumus sebagai berikut [10]

$$PSNR = 10 \log_{10} \frac{(2^n - 1)^2}{MSE} \quad (2\text{-}11)$$

dimana $$MSE = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (C_{ij} - R_{ij})^2 \quad (2\text{-}12)$$

dan $(2^n - 1)^2$ ialah nilai maksimum yang mungkin pada citra, dimana n ialah jumlah bit per sampel.

PSNR dapat dihitung secara mudah dan cepat serta sangat sering digunakan sebagai parameter kualitas dalam kompresi video. Nilai PSNR kecil menunjukkan kualitas rendah dari hasil rekonstruksi kompresi, sedangkan nilai PSNR besar menunjukkan kualitas tinggi dari hasil rekonstruksi kompresi. Nilai PSNR tidak selalu berkorelasi dengan kualitas subjektif dari hasil rekonstruksi. Pada Gambar 2.25 menunjukkan citra dengan kualitas bagian utama yang baik namun memiliki PSNR relatif rendah.

Gambar 2. 25 Citra dengan background kabur (PSNR 27.7 dB) [10]

2.7.2 Rasio Kompresi

Disamping melihat hasil rekonstruksi, rasio kompresi juga menjadi parameter penting yang menjadi ukuran performansi suatu teknik kompresi. Performansi suatu teknik kompresi yang baik, biasanya ditandai dengan nilai rasio kompresi yang tinggi. Rasio kompresi ialah perbandingan ukuran data sebelum dikompresi terhadap ukuran data sesudah dikompresi. Secara matematis dapat dijabarkan dengan persamaan berikut [25]

$$Cr = \frac{bits\ before\ \ compression}{bits\ after\ \ compression} \quad (2\text{-}13)$$

BAB 3
PERANCANGAN SIMULASI

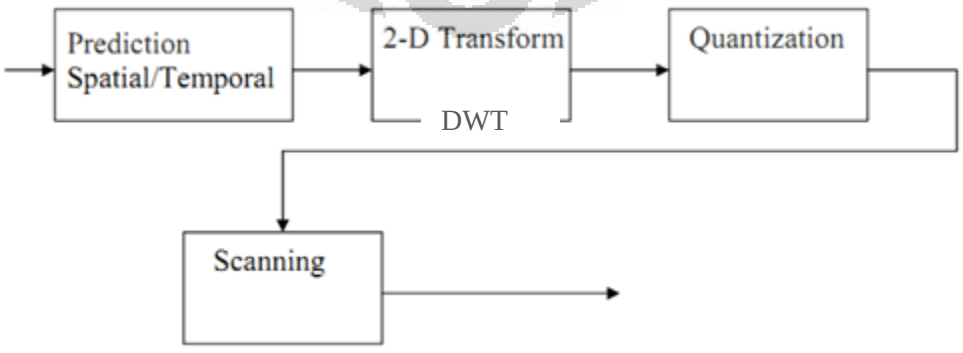
Perancangan simulasi kompresi video medis dengan H.264 berdasarkan transformasi wavelet diskrit dilakukan berdasarkan blok diagram pada referensi [23]. Pada bagian transformasi 2D, digunakan dua transformasi 2D yang berbeda yaitu DCT serta DWT. Disamping itu, untuk menyederhanakan kompleksitas maka tidak seluruh algoritma H.264 dilakukan.

3.1 Perancangan Simulasi

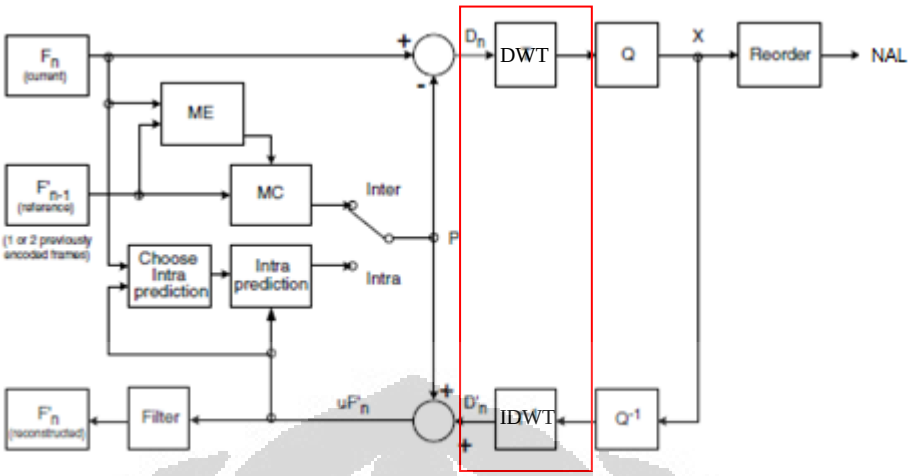
Simulasi kompresi video medis dengan H.264 yang diajukan mengacu pada algoritma *Baseline Profile* yaitu profil dengan fitur terendah dari standar H.264. *Baseline Profile* pada H.264 memiliki fitur [10]:

1. I-*slice*
2. P-*slice*
3. CAVLC
4. Prediksi intra hingga 4x4 *sub-macroblock*
5. Prediksi inter hingga 4x4 *sub-macroblock*

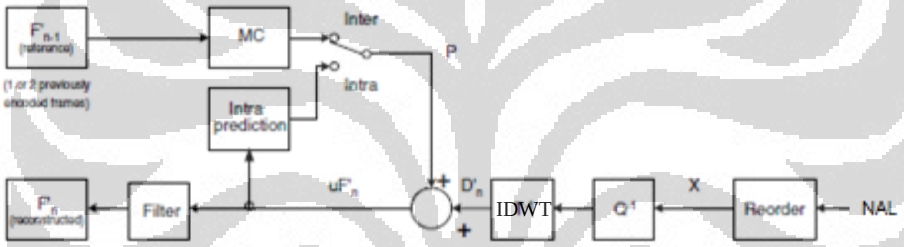
Gambar 3.1 di bawah ini adalah model perancangan simulasi yang diajukan. Sedangkan Gambar 3.2 dan Gambar 3.3 ialah arsitektur *encoder* dan *decoder* dari perancangan simulasi H.264 dengan DWT.



Gambar 3. 1 Diagram blok *Encode*r H.264 dengan DWT yang diajukan



Gambar 3. 2 Arsitektur *encoder* kompresi video medis berdasarkan algoritma H.264 dengan DWT



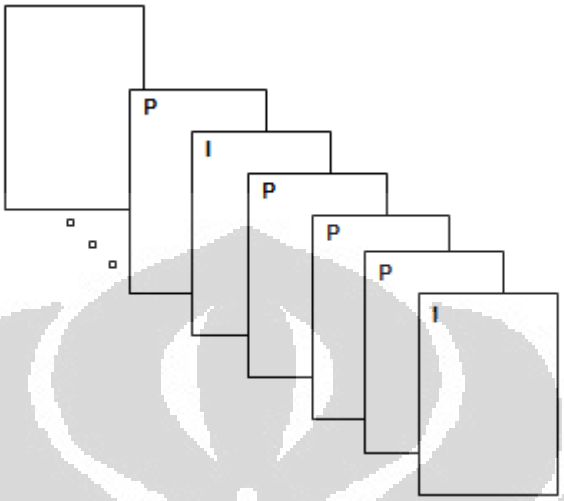
Gambar 3. 3 Arsitektur *decoder* kompresi video medis berdasarkan algoritma H.264 dengan DWT

Secara umum, arsitektur pada Gambar 3.2 dan Gambar 3.3 dapat disederhanakan menjadi blok-blok, dimana *encoder* terdiri dari prediksi secara spasial dan temporal, transformasi 2-D, kuantisasi, *scanning* koefisien, serta *entropy coding* seperti Gambar 3.1. Pada *decoder* dilakukan rekonstruksi dari data hasil *encode* dengan *inverse* zigzag *scanning*, *rescaling*, *inverse* transformasi 2D, serta prediksi spasial dan temporal.

3.1.1 Prediksi Spasial dan Temporal

Baseline Profile pada H.264 hanya mendukung adanya *slice* I dan P dimana setiap *frame* dapat terdiri dari satu *slice* atau lebih. *Slice* I akan diprediksi secara spasial dan *slice* P diprediksi secara temporal dengan referensi berupa *slice* yang dikodekan sebelumnya. Pada simulasi yang diajukan, setiap *frame* hanya

terdiri dari satu *slice* sehingga digunakan skema *Group of Pictures* (GOP) IPPP seperti yang juga digunakan pada referensi [3]. Skema GOP pada simulasi ditunjukkan oleh Gambar 3.4.



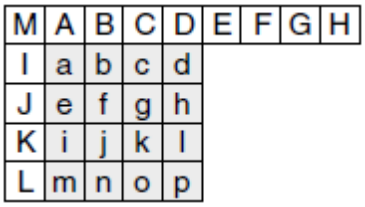
Gambar 3. 4 Skema *Group of Pictures* yang digunakan

3.1.1.1 Prediksi Spasial (*Frame I*)

Frame pertama atau *frame I* dari setiap GOP akan diproses dengan mode *intraframe*. Pada mode *intraframe*, citra akan dibagi ke dalam *macroblock* 16x16. Prediksi spasial dapat menggunakan *sub-macroblock* yang lebih kecil yaitu 8x8 dan 4x4.

Selanjutnya dilakukan tahap prediksi spasial yaitu prediksi dengan mode yang tersedia sesuai dengan ukuran *sub-macroblock* yang digunakan. Untuk prediksi dengan *sub-macroblock* 8x8 terdapat empat mode seperti pada Gambar 2.20. Sedangkan untuk prediksi dengan *sub-macroblock* 4x4 terdapat delapan mode seperti pada Gambar 2.21.

Pada simulasi yang dilakukan, hanya dilakukan prediksi spasial dengan mode DC pada *sub-macroblock* berukuran 4x4. Pada tahapan prediksi spasial ini, maka nilai piksel dalam suatu *sub-macroblock* akan diprediksi dengan nilai piksel tepi dari *sub-macroblock* yang bersebelahan. Nilai piksel yang diprediksi dihitung dengan mode DC atau dihitung rata-rata nilai piksel yang bersebelahan seperti pada Gambar 3.5.



Gambar 3. 5 Prediksi spasial pada *sub-macroblock* berukuran 4x4 [10]

Untuk menghitung nilai piksel a,b,c,d,e,...,p dengan mode DC, dilakukan perhitungan nilai total dari piksel yang bersebelahan dengan *sub-macroblock* tersebut, kemudian dirata-ratakan [26].

$$mean = (A+B+C+D+E+I+J+K+L+4)/8 \quad (3\text{-}1)$$

Kemudian nilai piksel a,b,c,d,e,...,p digantikan dengan nilai *mean*. Prediksi spasial dengan mode DC pada simulasi dilakukan secara seragam pada seluruh matriks *frame* I. Hal ini akan menyebabkan munculnya artifak pada bagian batas antara latar dengan bagian tubuh yang terekam.

Setelah didapatkan hasil prediksi spasial untuk seluruh *frame*, maka selanjutnya dihitung residu prediksi spasial dengan menghitung selisih *frame* I dengan hasil prediksi spasial.

3.1.1.2 Prediksi Temporal

Kemudian selain *frame* yang telah ditentukan sebagai *frame* I, *frame* akan diproses menjadi *frame* P. Pada *frame* P akan dilakukan prediksi temporal berbasis blok, dimana *frame* akan dibagi ke dalam blok-blok kemudian dicari masing-masing *motion vector* dengan metode pencarian *Three Step Search*.

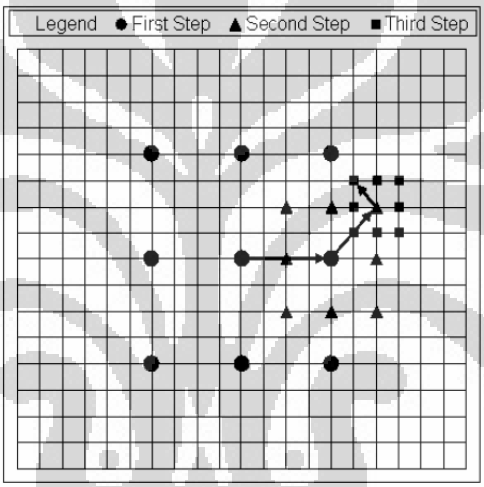
- *Motion vector*

Untuk *frame* P maka prediksi temporal akan dilakukan dengan mengambil referensi berupa *frame* I ataupun *frame* P sebelum *frame* yang akan dikodekan. *Motion vector* akan dihitung dengan metode *Three Step Search*, seperti pada Gambar 3.6, dimana dari delapan piksel tertentu dalam daerah

pencarian dilakukan perhitungan *Mean Absolute Difference* dengan persamaan [10]:

$$MAD = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|C_{ij} - R_{ij}\right| \quad (3\text{-}2)$$

Kemudian dicari nilai MAD terendah untuk selanjutnya dijadikan acuan untuk menghitung nilai delapan piksel di sekitarnya hingga iterasi sebanyak tiga kali. Nilai piksel dengan MAD terendah dari tiga iterasi tersebut dipilih sebagai *motion vector* yang diprediksi. Metode *Three Step Search* memiliki keunggulan waktu komputasi yang lebih kecil.



Gambar 3. 6 Pencarian *motion vector* dengan metode *Three Step Search* [24]

Dengan demikian, setiap *sub-macroblock* akan memiliki *motion vector* masing-masing.

Pada simulasi yang dilakukan, *motion vector* dari setiap *sub-macroblock* akan disimpan. Tidak dilakukan prediksi dari *motion vector* yang sudah dikodekan. Dengan demikian, setiap *frame* akan memiliki jumlah *motion vector* yang sangat banyak yang akan mempengaruhi ukuran data hasil *encode* dan rasio kompresi.

- *Motion compensation*

Setelah itu, *motion vector* diimplementasikan pada citra referensi untuk mendapatkan *motion compensation*. Kemudian dihitung residu dari hasil prediksi

temporal dengan menghitung selisih *frame* P dengan citra hasil *motion compensation*.

3.1.2 Transformasi 2D

Tahap transformasi 2D merupakan bagian utama yang akan ditelaah efeknya. Transformasi 2D yang akan dilakukan menggunakan transformasi DCT dan DWT. Transformasi dilakukan pada matriks residu baik dari hasil prediksi spasial maupun temporal.

Transformasi DCT pada simulasi yang dilakukan mengambil matriks DCT dari referensi [10] dengan ukuran *sub-macroblock* sebesar 4x4 sehingga dapat memperkecil efek munculnya artifak *blocking*. Matriks C berikut ini dengan ukuran 4x4 ialah matriks DCT yang digunakan pada simulasi.

$$C_f = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \quad (3\text{-}3)$$

dengan $a = \frac{1}{2}$; $b = \sqrt{\frac{1}{2}}\cos\frac{\pi}{8}$; $c = \sqrt{\frac{1}{2}}\cos\frac{3\pi}{8}$

Sedangkan ketika dilakukan penggantian transformasi dari DCT menjadi DWT, diuji beberapa filter DWT yang berbeda. Jenis wavelet induk *7/9 biorthogonal* dengan level dekomposisi sebanyak 3 pada referensi [6] diketahui memiliki gain 0,6 - 1,0 dB dibandingkan DCT. Namun pada simulasi ini, jenis wavelet induk yang digunakan ialah Haar, Daubechies, Biorthogonal, Symlet, dan Coiflet yang populer digunakan karena ketersediaan algoritma *fast transform* untuk lima filter tersebut. Dekomposisi dengan menggunakan filter DWT dilakukan dengan tiga level seperti referensi [6] dengan menggunakan fungsi *wavedec2* yang sudah tersedia dari MATLAB. Hasil dekomposisi dari DWT ialah berupa koefisien matriks *sub-band* yang tersusun dalam bentuk *tree-structured* dalam satu *array* serta ukuran matriks setiap *sub-band*.

3.1.3 Kuantisasi

Selanjutnya matriks koefisien hasil transformasi DCT maupun *array* koefisien DWT dikuantisasi dengan menggunakan kuantisasi skalar seragam dengan nilai parameter kuantisasi sebesar 36. Berdasarkan referensi [10] nilai

parameter kuantisasi 36 memiliki *step* kuantisasi sebesar 40 sehingga setiap koefisien hasil transformasi 2D akan dibagi dengan nilai *step* kuantisasi tersebut dengan persamaan [10].

$$Z_{ij} = round(\frac{Y_{ij}}{Qstep}) \quad (3\text{-}4)$$

Pada kedua jenis transformasi 2D dilakukan kuantisasi dengan nilai parameter kuantisasi yang sama agar dapat dibandingkan hasilnya.

3.1.4 *Scanning*

Setelah dilakukan kuantisasi, maka koefisien-koefisien hasil transformasi 2D akan dis*canning* dengan menggunakan *zig-zag scanning*. Hal ini bertujuan untuk mengelompokkan koefisien yang tidak bernilai nol di bagian awal dan koefisien yang bernilai nol di bagian akhir sehingga mudah dilakukan *entropy coding*.

Pada koefisien hasil transformasi DWT tidak dilakukan *scanning* lagi sebab fungsi dekomposisi wavelet yang digunakan sudah menghasilkan koefisien *sub-band* yang tersusun secara *tree-structured*. Susunan *tree-structured* pada hasil dekomposisi wavelet memiliki struktur koefisien *sub-band* frekuensi rendah LL yang paling merepresentasikan citra akan berada di urutan awal dan *sub-band* dengan frekuensi tinggi yang bernilai rendah yakni pada *sub-band* HH di urutan akhir dapat dikodekan dengan *entropy coding* secara efisien karena sudah terkelompok.

3.1.5 *Entropy coding*

Urutan koefisien hasil scanning pada standar H.264 *Baseline Profile* dikodekan dengan *entropy coding* CAVLC. CAVLC memanfaatkan korelasi antar *macroblock* yang terdapat dalam satu *slice* sehingga jumlah bit yang dibutuhkan untuk setiap koefisien tidak hanya bergantung dari probabilitas kemunculannya saja, tetapi juga bergantung dari jumlah bit pada *macroblock* yang sebelumnya sudah dikodekakn.

Namun demikian, pada simulasi yang dilakukan, tidak dilakukan *entropy coding* sebab perbandingan ukuran data hasil *encode* sudah dapat dilakukan tanpa melalui tahapan *entropy coding*.

3.2 Data Video Medis

Video medis yang digunakan berupa sekumpulan citra hasil rekam MRI dan CT dari beberapa bagian tubuh yang berbeda. Citra yang membentuk *frame* data video medis memiliki format RAS yang merupakan format citra untuk RGB maupun *grayscale* tanpa kompresi. Sekumpulan citra yang membentuk video medis MRI dan CT ini didapat dari situs http://www.cipr.rpi.edu/. Situs tersebut merupakan situs grup riset yang berkaitan dengan pemrosesan citra digital (*Center for Image Processing Research*) di *Electrical, Computer, and Systems Engineering Department,* Rensselaer Polytechnic Institute yang berlokasi di New York, Amerika Serikat.

Situs ini menyediakan daftar proyek dan publikasi yang pernah mereka lakukan, serta menyediakan *resource* data yang mereka gunakan dalam proyek tersebut baik yang berupa urutan citra maupun citra tunggal.

3.3 *Software* Simulasi

Simulasi akan dilakukan dengan menggunakan perangkat lunak Matlab versi R2010b. Perangkat lunak ini mampu membaca berbagai format citra maupun video termasuk format citra RAS dari data yang digunakan untuk simulasi. Disamping itu, perangkat lunak MATLAB juga memiliki fungsi yang memudahkan pengguna untuk melakukan dekomposisi citra dengan wavelet.

3.4 Analisis Performansi

Analisis performansi dari simulasi H.264 dengan DCT dan DWT yang dilakukan adalah rasio kompresi serta PSNR. Hal ini dikarenakan untuk kompresi video medis harus mempertahankan nilai PSNR yang relatif tinggi agar tidak muncul artifak yang menyebabkan kesalahan diagnosis. Sementara itu proses kompresi yang baik mengharuskan rasio kompresi yang tinggi sehingga tidak ada lagi data redundan.

Nilai PSNR akan dihitung untuk setiap *frame* yang disajikan dalam bentuk grafik dengan persamaan (2-11). Juga dihitung rasio kompresi dari total ukuran video medis sebelum dan sesudah dikodekan dengan persamaan (2-13).

3.4 Skenario Simulasi

Skenario simulasi ini adalah untuk menentukan nilai rasio kompresi dan PSNR dari video medis yang merekam beberapa bagian tubuh yang berbeda dengan data berupa sekumpulan citra berurutan hasil MRI dan CT. Skenario simulasi yang digunakan adalah:

1. Simulasi kompresi video medis dengan algoritma H.264 berdasarkan DCT dan DWT dengan beberapa filter atau wavelet induk tanpa mengubah parameter kompresi apapun kecuali transformasi 2D saja.
2. Hasil *encode* disimpan dalam bentuk data MATLAB dengan extension .mat digunakan untuk menghitung rasio kompresi ; sementara hasil *decode* disimpan dalam bentuk video digunakan untuk menghitung PSNR.
3. Pembandingan nilai PSNR dari algoritma kompresi H.264 berdasarkan DCT terhadap DWT dengan beberapa jenis filter untuk menguji ketepatan penggunaan transformasi wavelet diskrit pada tahap transformasi 2D secara *general*.
4. Pembandingan nilai rasio kompresi dari algoritma kompresi H.264 berdasarkan DCT terhadap DWT dengan beberapa jenis filter untuk menguji ketepatan penggunaan transformasi wavelet diskrit pada tahap transformasi 2D secara *general*.

BAB 4
HASIL SIMULASI DAN ANALISIS

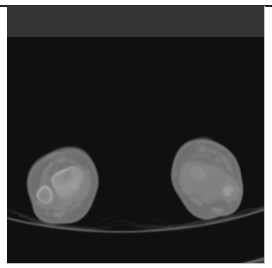
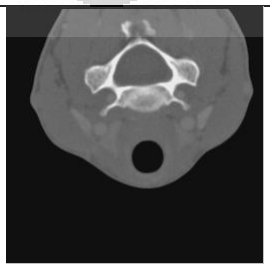
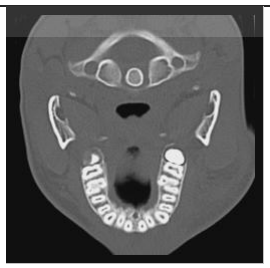
Simulasi dilakukan terhadap data medis yang telah disebutkan pada bagian 3.2 yaitu kumpulan citra MRI dan CT yang dibuat menjadi video. Video tersebut di-*encode* dan di-*decode* dengan algoritma H.264 *intraframe* dan *interframe* baik berdasarkan DCT dan DWT. Kemudian disimpan hasil rekonstruksinya dalam bentuk video dan dihitung PSNR serta rasio kompresi.

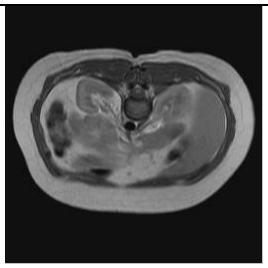
Seperti yang telah dijelaskan pada bagian 3.1 dan sub-bab yang terdapat di dalamnya, simulasi yang dilakukan mengikuti algoritma H.264, namun tidak seluruh fitur yang ada digunakan dalam simulasi ini.

4.1 Hasil Simulasi

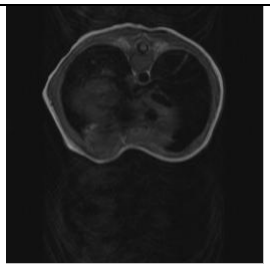
Hasil simulasi dari algoritma yang telah dibuat ialah berupa video rekonstruksi data yang telah di-*encode*. Video rekonstruksi tersebut kemudian dibandingkan dengan video orisinalnya untuk mendapatkan nilai PSNR setiap *frame* dalam video rekonstruksi serta nilai rasio kompresi file video rekonstruksi. Nilai PSNR dan rasio kompresi tidak dibandingkan antara data video yang berlainan karena berisi rekaman bagian tubuh yang berbeda dan memiliki karakteristik redundansi spasial dan temporal berbeda. Tabel 4.1 menunjukkan potongan *frame* dari video yang disimulasikan.

Tabel 4. 1 Potongan *frame* ke-12 dari masing-masing video yang disimulasikan

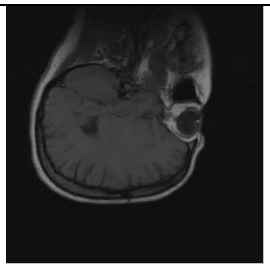
		
CT aperts	CT carotid	CT skull



MR liver



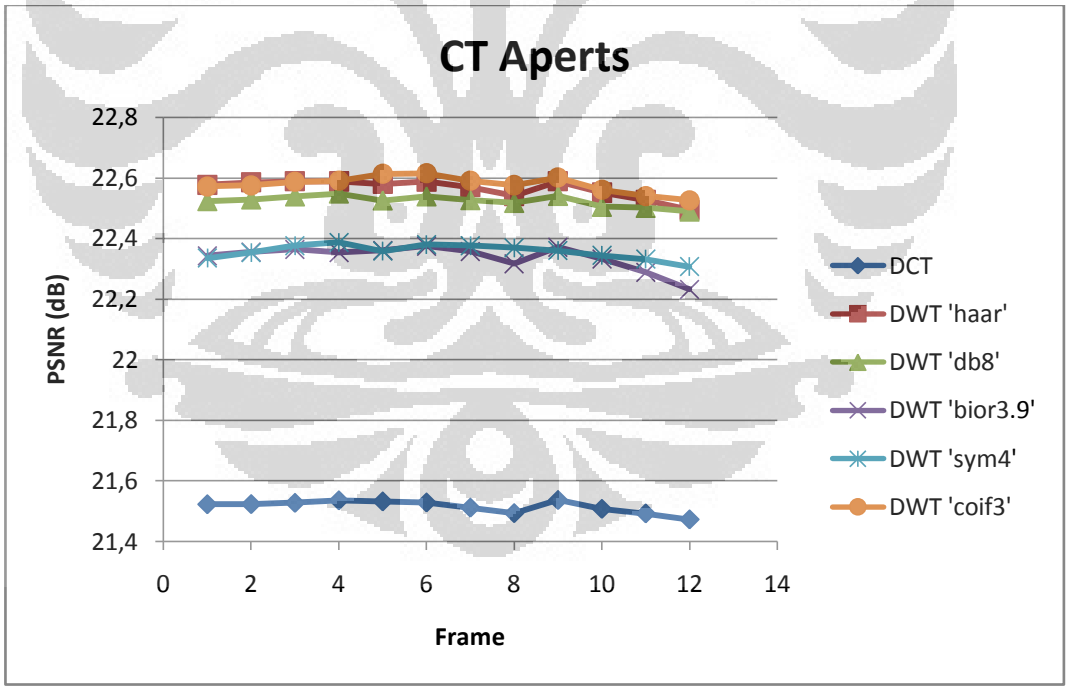
MR chest



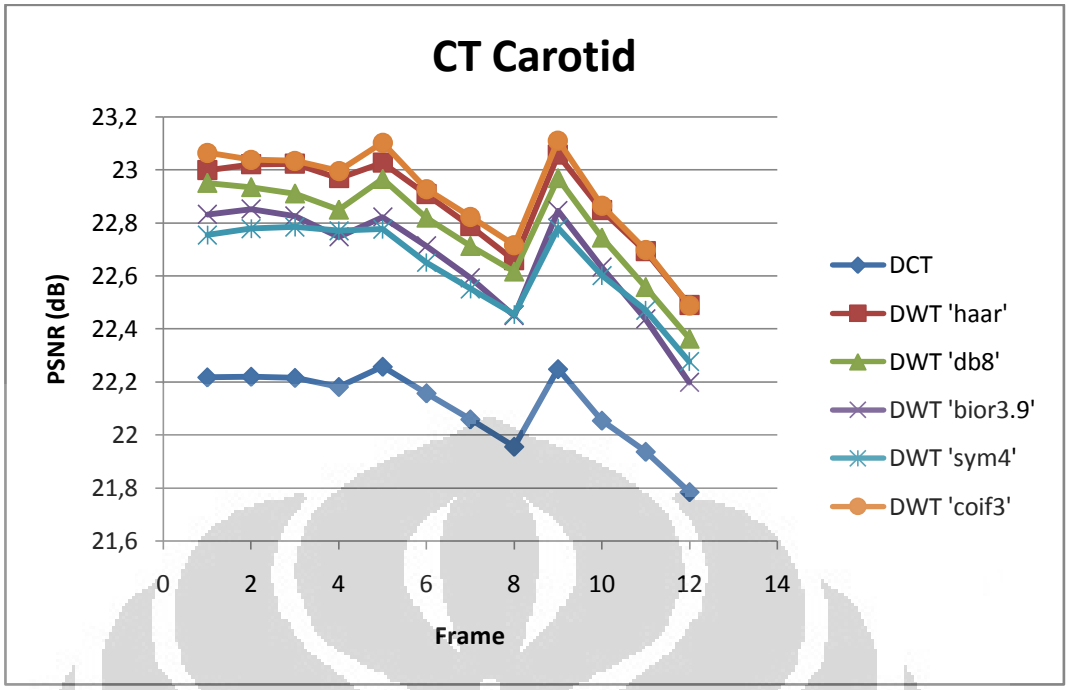
MR head

4.1.1 PSNR

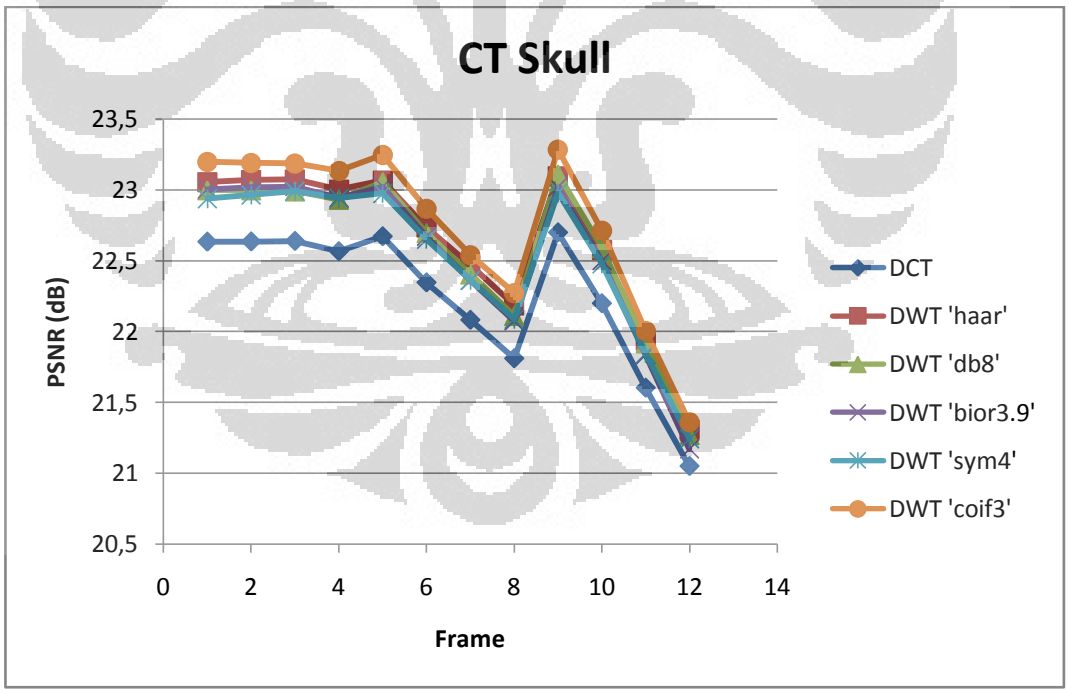
Video rekonstruksi dari hasil simulasi DCT dan DWT dengan filter Haar, Daubechies, Coiflet, Symlet dan Biorthogonal dihitung MSE-nya terhadap video orisinal untuk mendapatkan nilai PSNR setiap *frame*. Gambar 4.1 hingga Gambar 4.6 menunjukkan nilai PSNR untuk setiap *frame* dari masing-masing video yang disimulasikan baik dengan H.264 berdasarkan DCT maupun DWT dengan filter Haar, Daubechies, Coiflet, Symlet, dan Biorthogonal.



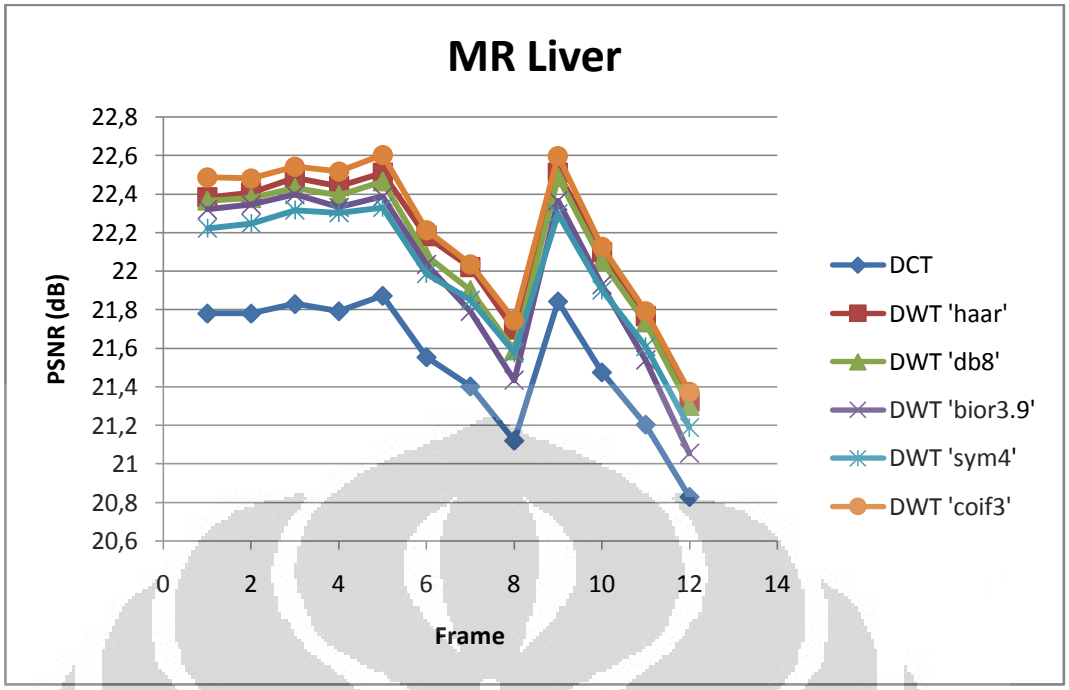
Gambar 4. 1 Grafik PSNR untuk setiap *frame* video CT Aperts berdasarkan DCT dan DWT



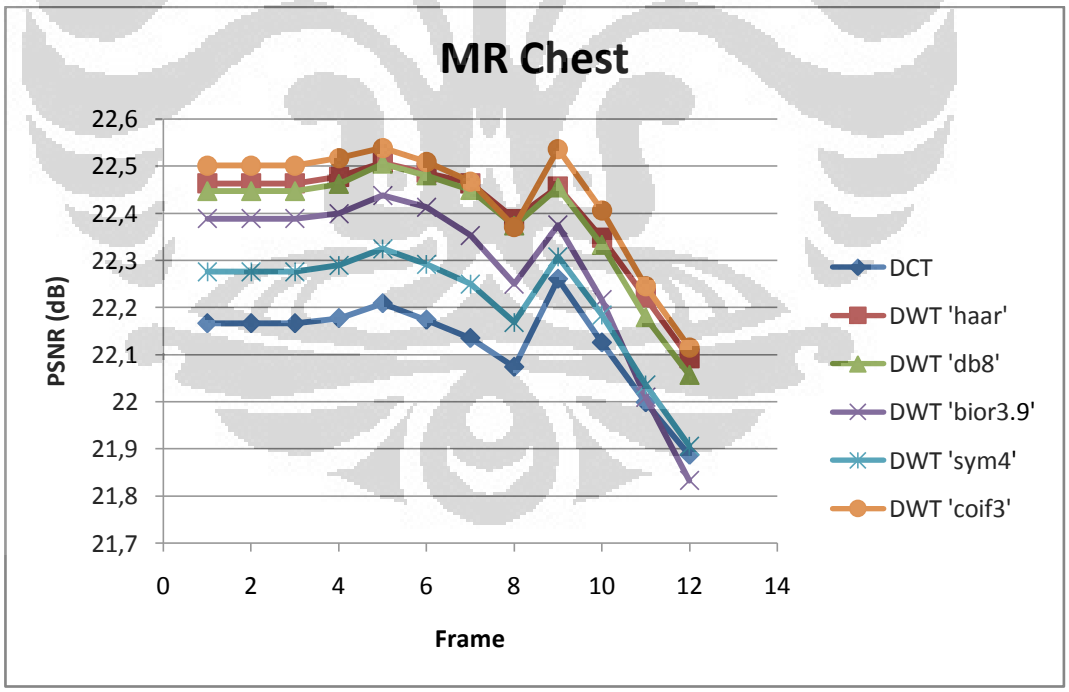
Gambar 4. 2 Grafik PSNR untuk setiap *frame* video CT Carotid berdasarkan DCT dan DWT



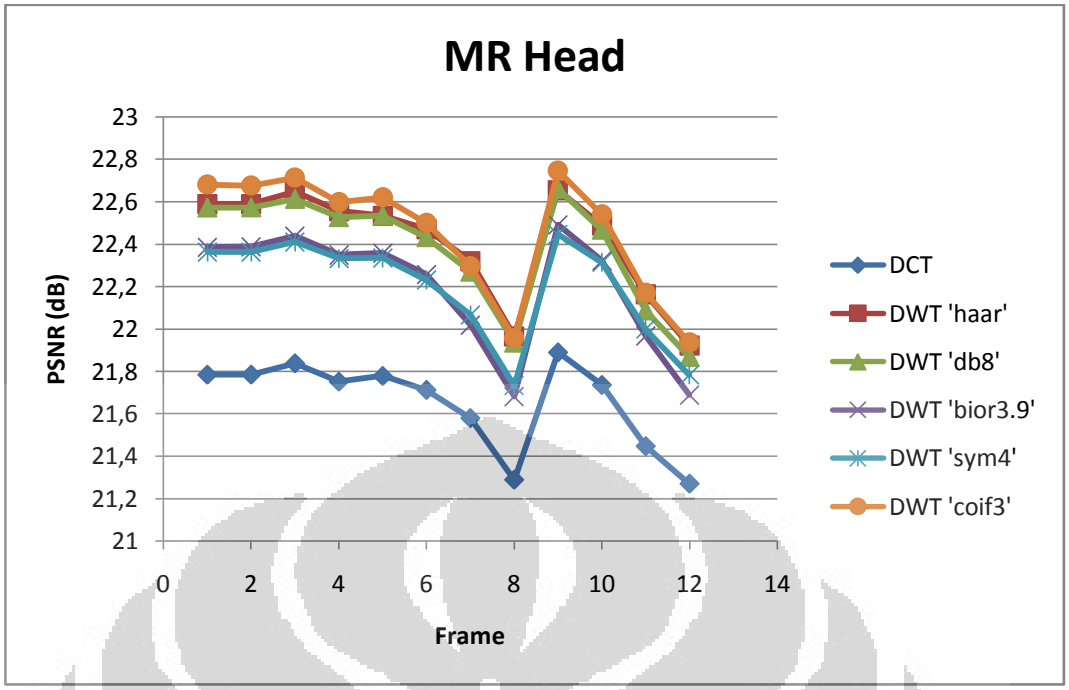
Gambar 4. 3 Grafik PSNR untuk setiap *frame* video CT Skull berdasarkan DCT dan DWT



Gambar 4. 4 Grafik PSNR untuk setiap *frame* video MR Liver berdasarkan DCT dan DWT



Gambar 4. 5 Grafik PSNR untuk setiap *frame* video MR Chest berdasarkan DCT dan DWT



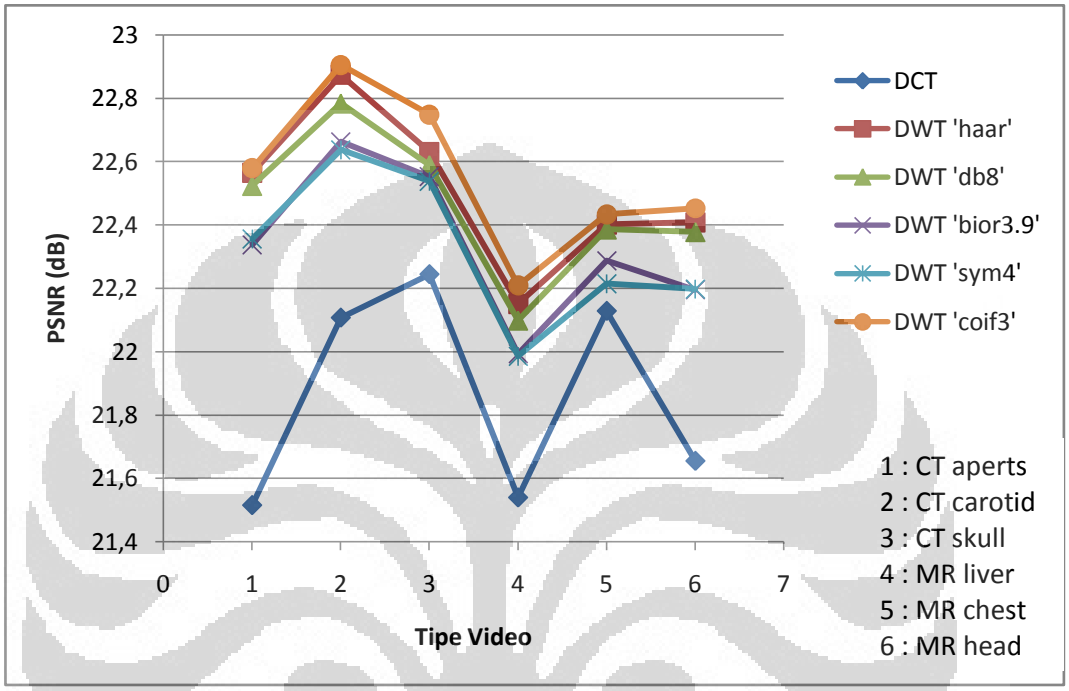
Gambar 4. 6 Grafik PSNR untuk setiap *frame* video MR Head berdasarkan DCT dan DWT

Nilai PSNR setiap *frame* dari masing-masing video tersebut dirata-ratakan dan hasilnya disajikan pada Tabel 4.2.

Tabel 4. 2 Nilai PSNR untuk tipe video berbeda berdasarkan simulasi H.264 dengan DCT dan DWT

Tipe video	Rata-rata PSNR					
	DCT	DWT 'haar'	DWT 'db8'	DWT 'sym4'	DWT 'coif3'	DWT 'bior3.9'
CT aperts	21,52	22,57	22,52	22,36	22,58	22,34
CT carotid	22,11	22,87	22,78	22,63	22,91	22,66
CT skull	22,24	22,63	22,59	22,54	22,75	22,55
MR liver	21,54	22,15	22,10	21,99	22,21	21,99
MR chest	22,13	22,40	22,39	22,22	22,43	22,29
MR_head	21,66	22,41	22,38	22,20	22,45	22,20

Dari Tabel 4.2 dapat dilihat bahwa kisaran nilai PSNR video rekonstruksi berada pada nilai 21-22 dB. Gambar 4.7 menunjukkan grafik perbandingan nilai PSNR dari Tabel 4.2.



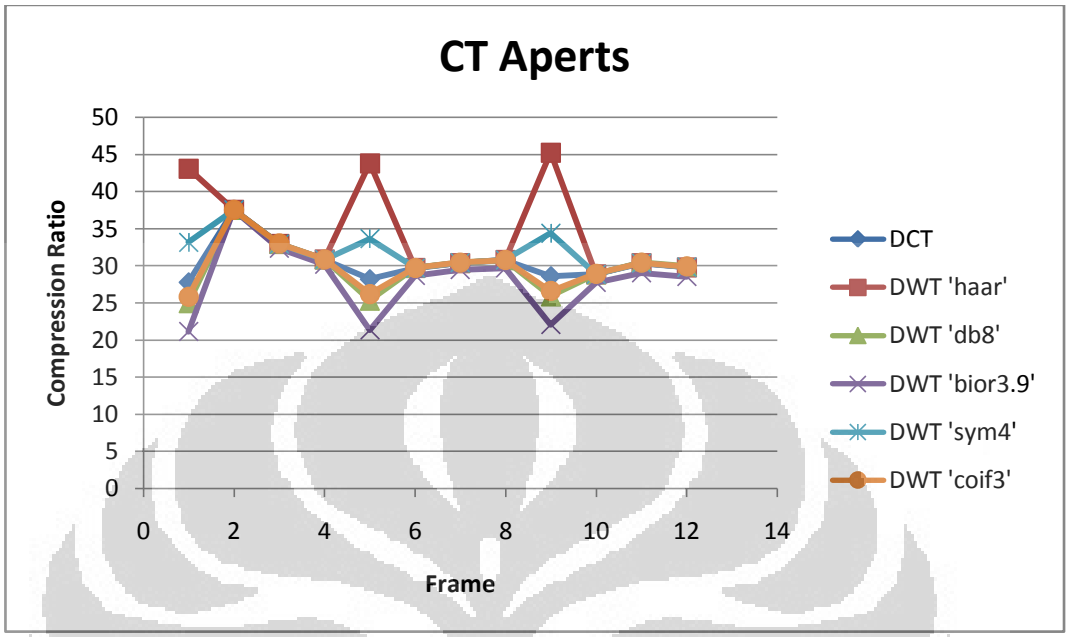
Gambar 4. 7 Grafik PSNR untuk tipe video yang berbeda dengan parameter kompresi sama

4.1.2 Rasio Kompresi

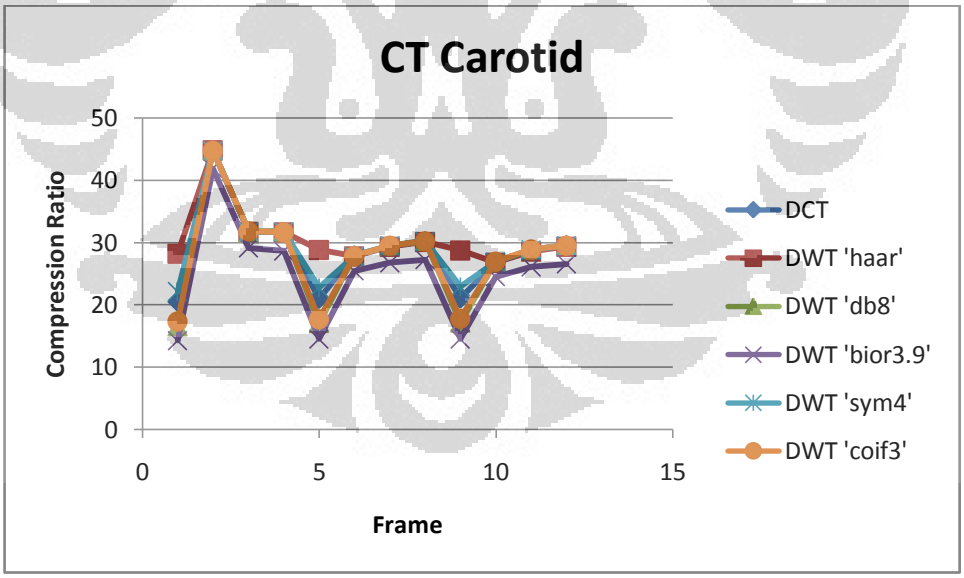
Rasio kompresi dilakukan dengan membandingkan jumlah bit yang diperlukan untuk membentuk video orisinal terhadap jumlah bit pada data hasil *encode*. Semakin besar rasio kompresi maka kompresi dianggap semakin baik karena telah melakukan pengurangan data redundan. Untuk menghitung rasio kompresi, jumlah bit pada data hasil *encode* tidak dihitung dari ukuran *file* melainkan dengan menghitung jumlah koefisien pada data *encode* yang tidak bernilai nol kemudian mengalikan jumlahnya dengan 8 bit.

Nilai rasio kompresi untuk setiap *frame* dari video yang diujikan ditampilkan pada Gambar 4.8 hingga Gambar 4.13. Sedangkan data rasio kompresi yang disajikan pada data Tabel 4.3 dan Gambar 4.14 ialah nilai rasio kompresi dari keseluruhan *file* hasil *encode*. Seluruh data rasio kompresi yang ditampilkan pada grafik dan tabel bukanlah nilai rasio kompresi sebenarnya dari

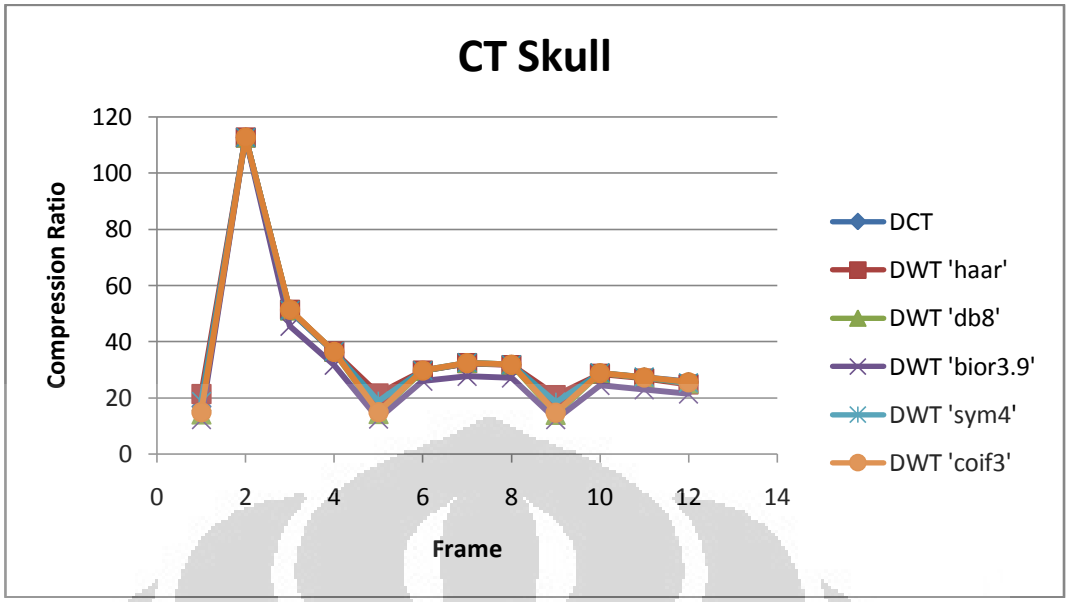
algoritma H.264 melainkan hanya prediksi saja karena tidak dilakukan algoritma H.264 secara penuh.



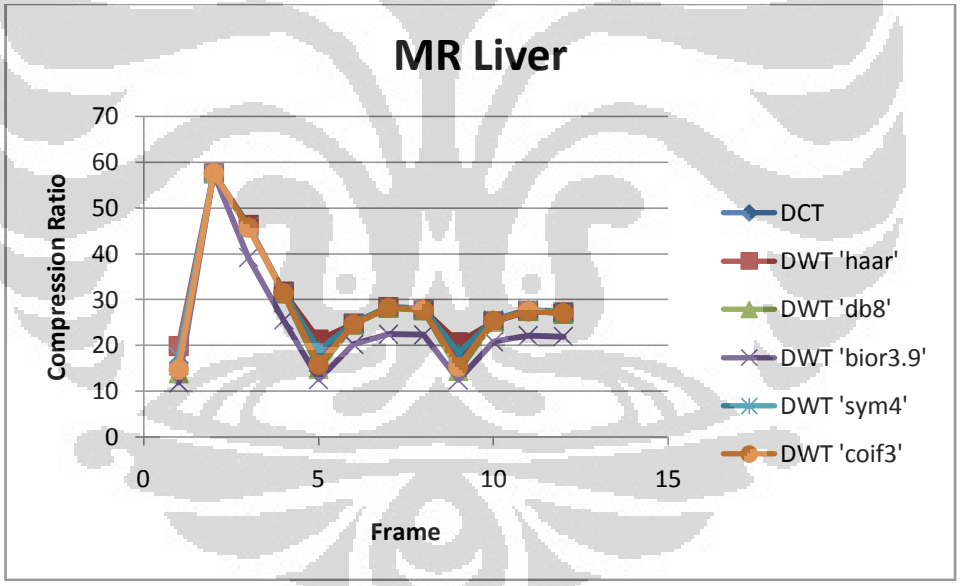
Gambar 4. 8 Grafik rasio kompresi untuk setiap *frame* video CT Aperts berdasarkan DCT dan DWT



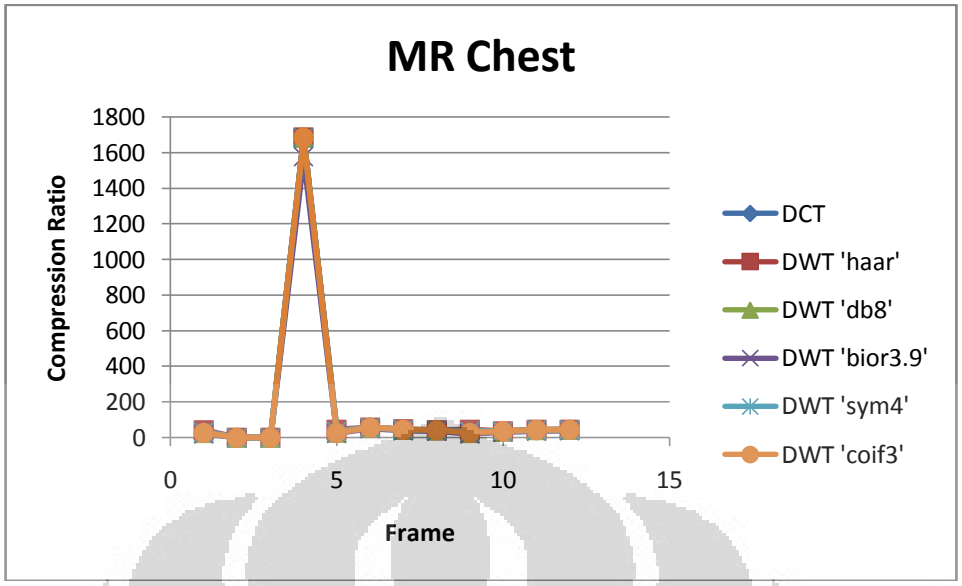
Gambar 4. 9 Grafik rasio kompresi untuk setiap *frame* video CT Carotid berdasarkan DCT dan DWT



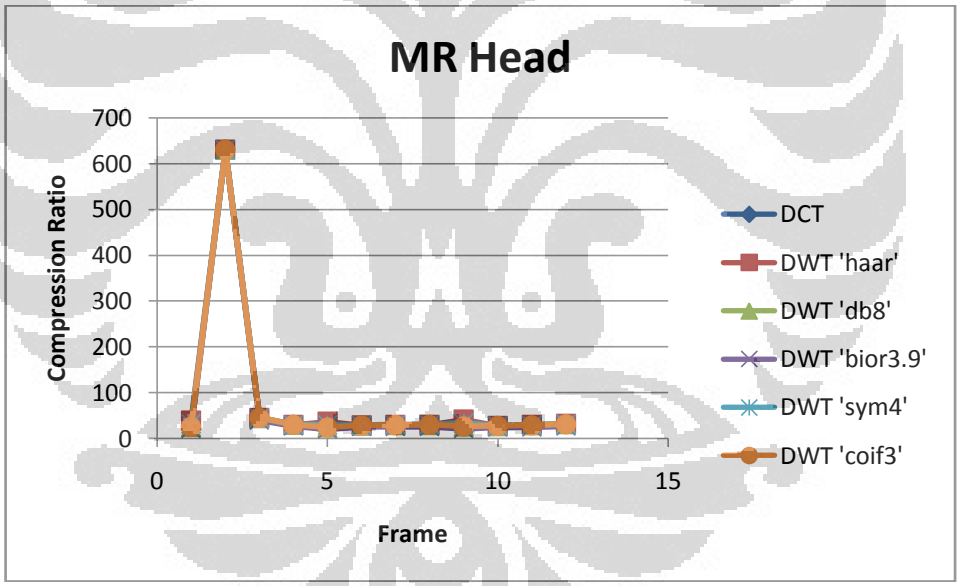
Gambar 4. 10 Grafik rasio kompresi untuk setiap *frame* video CT Skull berdasarkan DCT dan DWT



Gambar 4. 11 Grafik rasio kompresi untuk setiap *frame* video MR Liver berdasarkan DCT dan DWT



Gambar 4. 12 Grafik rasio kompresi untuk setiap *frame* video MR Chest berdasarkan DCT dan DWT

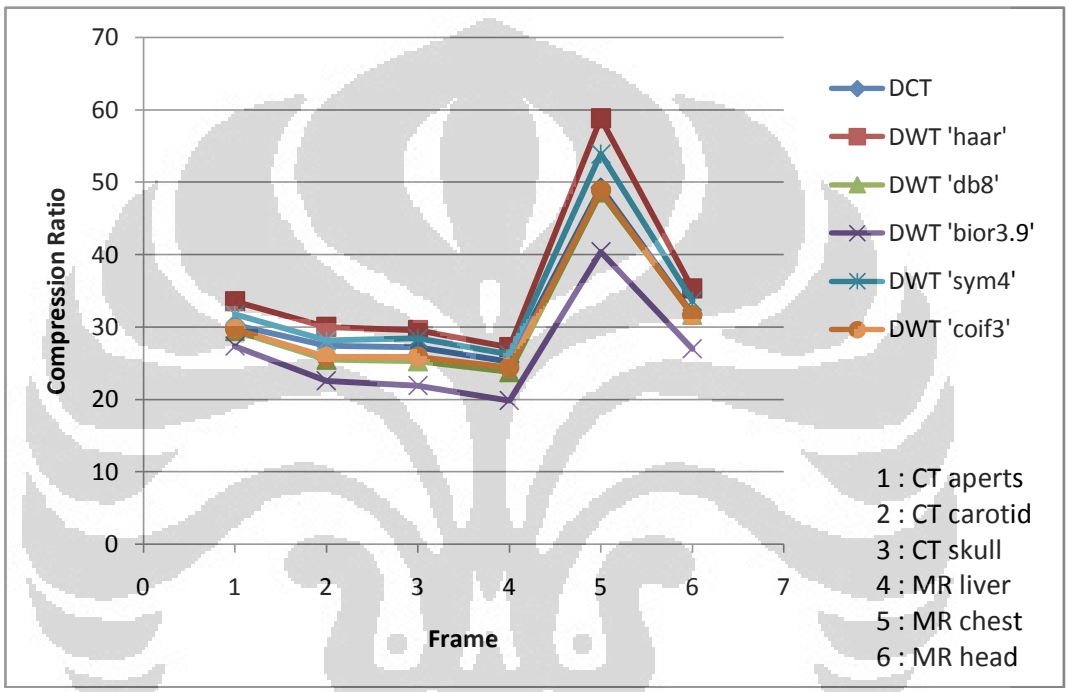


Gambar 4. 13 Grafik rasio kompresi untuk setiap *frame* video MR Head berdasarkan DCT dan DWT

Tabel 4. 3 Nilai rasio kompresi untuk tipe video berbeda berdasarkan simulasi H.264dengan DCT dan DWT

Tipe video	Rasio Kompresi					
	DCT	DWT ‘haar’	DWT ‘db8’	DWT ‘sym4’	DWT ‘coif3’	DWT ‘bior3.9’

CT aperts	30,29	33,54	29,46	31,71	29,71	27,39
CT carotid	27,39	30,01	25,50	28,13	25,86	22,60
CT skull	27,22	29,53	25,29	28,39	25,87	21,89
MR liver	25,21	27,20	23,78	26,21	24,41	19,84
MR chest	49,27	58,81	48,59	53,92	48,86	40,41
MR_head	31,78	35,27	31,70	33,61	31,63	27,00



Gambar 4. 14 Grafik rasio kompresi untuk tipe video berbeda berdasarkan simulasi dengan DCT dan DWT

4.2 Analisis

4.2.1 PSNR

Pada grafik pada Gambar 4.7 maupun Tabel 4.2 yang menunjukkan nilai rata-rata PSNR video rekonstruksi dengan DCT dan DWT, dapat dilihat bahwa nilai PSNR video rekonstruksi dengan simulasi DCT memiliki nilai terendah yakni berkisar antara 21,52-22,24 dB. Sedangkan video rekonstruksi dengan simulasi DWT dengan lima filter berbeda, seluruhnya memiliki nilai PSNR melebihi DCT pada kisaran 21,99-22,91 dB. Baik hasil yang ditampilkan pada tabel maupun grafik membuktikan bahwa DWT dapat diaplikasikan pada algoritma kompresi video H.264 yang memiliki *motion compensation* berbasis

blok 4x4 karena menghasilkan nilai PSNR yang lebih baik daripada DCT. Sebelumnya, DWT juga pernah diaplikasikan pada algoritma kompresi citra pada referensi [6] dan menghasilkan kualitas rekonstruksi citra yang lebih baik.

PSNR DCT terlihat cukup rendah dibandingkan dengan PSNR DWT dengan lima filter disebabkan oleh proses DCT yang dilakukan dalam blok-blok kecil dari suatu *frame*. Artifak *blocking* yang terjadi baik pada bagian yang homogen maupun pada batas antara latar dengan objek menyebabkan error yang semakin besar pada hasil rekonstruksi. Sedangkan proses dekomposisi wavelet tidak dilakukan dalam blok-blok melainkan mengolah keseluruhan citra sehingga tidak muncul akibat artifak *blocking*. Di samping itu, fitur dekomposisi wavelet 2D memberikan *padding* dengan tipe tertentu pada setiap *sub-band* hasil dekomposisi untuk mengatasi efek piksel tepi saat dilakukan dekomposisi [27]. Pada dasarnya, DWT memiliki skema sederhana yaitu melakukan konvolusi dengan filter lalu di-*downsampling*. Filter wavelet yang digunakan pada simulasi memiliki jumlah koefisien yang terbatas. Pada konvolusi dengan sinyal yang memiliki jumlah koefisien terbatas akan dapat menyebabkan distorsi piksel tepi *subband*. Untuk itulah secara sederhana ditambahkan piksel *padding* pada tepi matriks *sub-band* untuk mengatasi distorsi tepi tersebut. Nilai piksel *padding* akan membuat error pada rekonstruksi citra menjadi lebih kecil. Namun adanya alokasi *padding* ini akan mempengaruhi nilai rasio kompresi.

Gain yang dihasilkan dari penggunaan DWT pada simulasi yang dilakukan berkisar antara 0,09-1,06 dB. Nilai PSNR pada video rekonstruksi hasil simulasi ini dapat ditingkatkan dengan menggunakan *adaptive in-loop deblocking* filter. Sebab pada *encoder*, *frame* yang digunakan sebagai referensi untuk prediksi temporal adalah hasil rekonstruksi dari *frame* yang sudah di-*encode*. Dengan menggunakan *adaptive deblocking filter* yang diletakkan pada bagian *backward* (rekonstruksi) *encoder* maka akan mengurangi artifak *blocking* error pada *frame* yang dijadikan referensi tersebut sehingga dapat mencegah akumulasi error pada *frame* yang akan di-*encode* berikutnya.

Pemilihan lima jenis filter DWT yaitu Haar, Daubechies, Symlet, Coiflet, dan Biorthogonal didasari pada kepopuleran penggunaan filter tersebut pada pengolahan citra dan video digital karena kelima filter tersebut dibangun

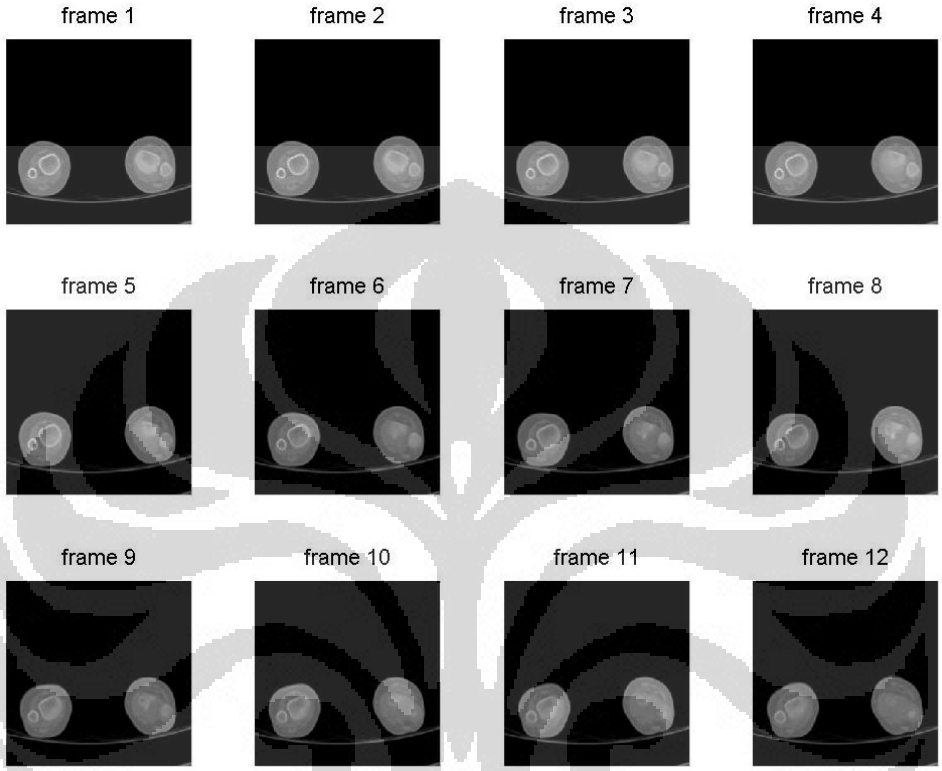
berdasarkan filter FIR dan memiliki *fast algorithm* yang mempercepat dan menyederhanakan komputasi serta implementasi.

Dari Tabel 4.2 dan Gambar 4.7 dapat terlihat bahwa DWT dengan filter berbeda menghasilkan nilai PSNR berbeda pula. Coiflet3 menghasilkan *gain* PSNR yang paling baik yaitu hingga 1,06 dB untuk video 'CT aperts' dibanding DCT. Nilai PSNR yang cukup tinggi disebabkan oleh filter Coiflet3 memiliki karakteristik mendekati simetri serta lebih *compact* karena memiliki banyak *zero moment*. Karakteristik *error* simetri lebih ditoleransi secara visual daripada error akibat asimetri serta lebih handal dalam menangani nilai tepi piksel. Dengan demikian Coiflet dapat menghasilkan nilai PSNR yang lebih tinggi.

PSNR dari DWT dengan filter 'bior3.9' memiliki nilai yang relatif lebih rendah daripada filter DWT lainnya. Hal ini disebabkan oleh karakteristik biorthogonal yang hanya mendekati orthonormal, dimana hal ini menyebabkan rekonstruksi *sub-band* hasil dekomposisi tidak sempurna. Disamping itu, di antara kelima filter yang digunakan, 'bior3.9' memiliki jumlah koefisien filter mencapai 20 [28]. Semakin panjang filter yang digunakan maka akan menambah distorsi berupa *ringing noise* pada rekonstruksi yang mudah dideteksi secara visual [29]. Sementara 'sym4' yang memiliki karakteristik hampir sama dengan 'coif3' ternyata juga memiliki nilai PSNR yang relatif rendah. Hal ini diakibatkan oleh karakteristiknya yang merepresentasikan suatu sinyal dengan basis fungsi wavelet yang saling berkaitan satu sama lainnya melalui fungsi translasi [30]. Karakteristik ini dapat menyebabkan distorsi pada rekonstruksi.

Pada simulasi yang dilakukan, video yang digunakan memiliki objek yang berbeda-beda. Perbedaan objek yang direkam ini menyebabkan perbedaan redundansi yang terdapat dalam video. Dengan demikian, jika dilakukan kompresi dengan parameter yang sama, maka hasil rekonstruksinya pun menjadi berbeda. Pada Gambar 4.1 yang menunjukkan grafik PSNR untuk setiap *frame* video CT Aperts, terlihat bahwa perubahan PSNR tidak signifikan seiring dengan perubahan *frame*. Hal ini disebabkan oleh akumulasi error yang rendah pada video hasil rekonstruksi. Akumulasi error dapat terjadi akibat prediksi spasial yang mengenai batas antara objek dan latar sehingga piksel batas tersebut menjadi *blur*. Di samping itu, perubahan temporal yang besar juga memungkinkan *error* akibat

tidak tepatnya *frame* kompensasi yang dihasilkan oleh *motion vector*. Pada video CT Aperts yang ditunjukkan pada Gambar 4.15, terlihat bahwa perubahan temporal antar *frame* tidak signifikan sehingga akumulasi *error* yang besar tidak terjadi. Dengan demikian, didapat nilai PSNR yang tidak berubah-ubah drastis.



Gambar 4. 15 Video CT aperts dalam potongan *frame*

Sedangkan pada Gambar 4.2 hingga Gambar 4.7, terlihat bahwa nilai PSNR pada *frame* ke-9 mengalami lonjakan. Pada *frame* ke-9 dilakukan prediksi spasial sehingga rekonstruksinya tidak mengikutsertakan akumulasi error dari *frame-frame* sebelumnya. Kemudian penurunan nilai PSNR pada *frame-frame* selanjutnya yang cukup besar disebabkan oleh perubahan temporal yang semakin signifikan dimana *error* kompensasinya ikut terakumulasi karena mengambil *frame* referensi dari *frame* sebelumnya.

Dari keseluruhan Gambar 4.1 hingga Gambar 4.7 terlihat bahwa nilai PSNR pada *frame* 1,5, dan 9 selalu menjadi maksimum lokal. Hal ini diakibatkan oleh skema GOP yang digunakan ialah IPPP dimana skema tersebut berulang di sepanjang jumlah *frame* video. Hasil prediksi spasial atau *intraframe* lebih baik

daripada prediksi temporal karena tidak menggunakan *frame* sebelumnya yang terakumulasi *error* sebagai referensi.

4.2.2 Rasio Kompresi

Rasio kompresi menunjukkan seberapa besar pengurangan redundansi baik spasial maupun temporal pada suatu video. Nilai rasio kompresi yang besar menunjukkan bahwa redundansi sudah banyak dihilangkan sehingga data dapat disimpan dan dikirimkan dengan lebih efisien. Dilakukan perhitungan jumlah koefisien yang tidak bernilai nol dari data hasil *encode* pada simulasi, kemudian dikalikan dengan 8 bit. Dengan demikian, didapat perkiraan ukuran jumlah bit yang dibutuhkan untuk data hasil *encode* yang harus dikirimkan atau disimpan.

Untuk *frame* I, data yang digunakan pada *decoder* dan dihitung sebagai data hasil *encode* ialah hasil prediksi intra*frame* dan residunya. Sedangkan pada *frame* P, data yang dihitung hanya residu dari prediksi temporal dan *motion vector*nya. Setelah didapat jumlah bit yang merepresentasikan setiap *frame* I dan P, maka dihitung rasio kompresi untuk setiap *frame*. Hasilnya ditunjukkan pada Gambar 4.8 hingga Gambar 4.13.

Pada Tabel 4.3 dan Gambar 4.2 terlihat bahwa rasio kompresi dari DCT dan DWT tidak jauh berbeda. Secara umum, jika melihat tren dari keseluruhan nilai rasio kompresi DWT, maka dapat dikatakan bahwa rasio kompresi DWT lebih baik daripada DCT. Ditambah lagi, berdasarkan referensi [27] diketahui bahwa pada dekomposisi wavelet 2D terdapat piksel padding pada matriks *sub-band* hasil dekomposisi. Hal ini menyebabkan jumlah koefisien yang tidak bernilai nol akan semakin banyak dan butuh lebih banyak bit untuk mewakili data hasil *encode*.

Filter ‘bior3.9’ menghasilkan lebih banyak piksel *padding* pada matriks *sub-band* karena karakteristiknya yang hanya mendekati orthonormal dan kurang *compact* sehingga membutuhkan lebih banyak piksel padding untuk mengatasi distorsi tepi *sub-band*. Oleh karena itu, jumlah bit yang merepresentasikan data hasil *encode* pun semakin banyak sehingga rasio kompresi menjadi menurun.

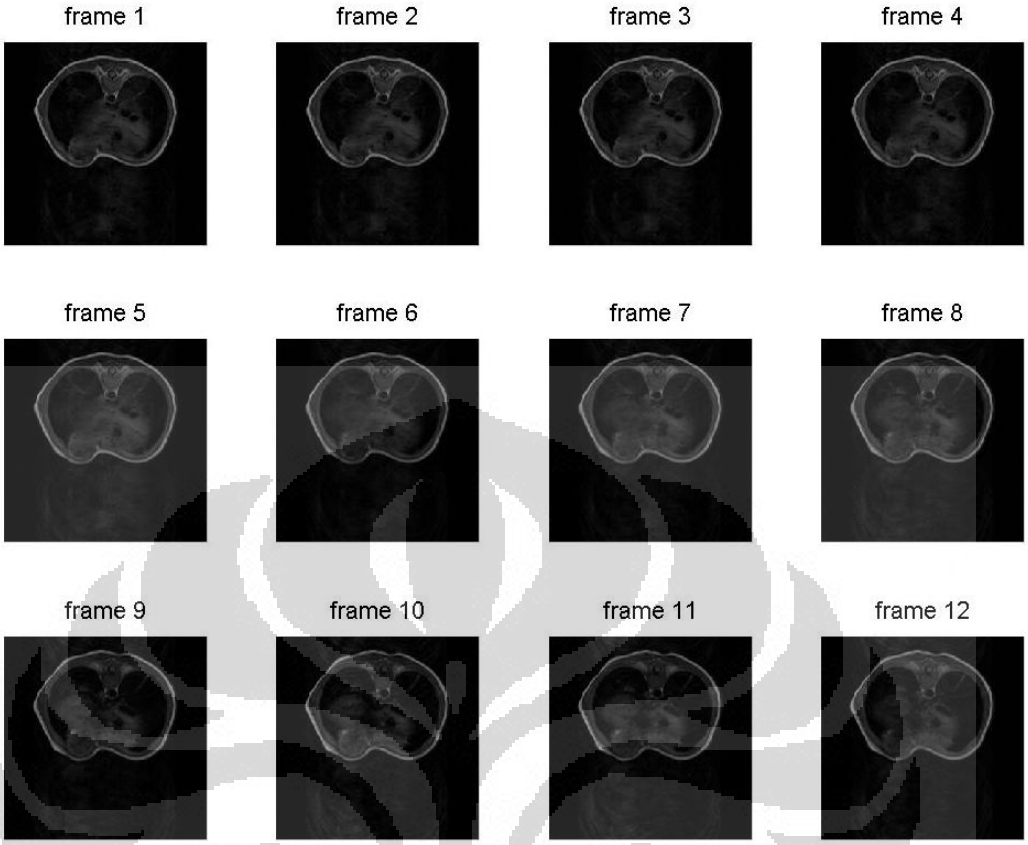
Nilai rasio kompresi ini juga dapat menjadi lebih besar untuk setiap video jika dilakukan *entropy coding* yang sesuai. Misalkan dengan menerapkan CAVLC

maupun *Huffman Code* yang mengkodekan setiap simbol atau koefisien dengan jumlah bit yang berbeda tergantung dari probabilitas kemunculannya. Penerapan *entropy coding* ini akan meningkatkan rasio kompresi dibandingkan dengan hasil prediksi yang dilakukan dimana diasumsikan setiap koefisien yang tidak bernilai nol direpresentasikan dengan 8 bit.

Untuk setiap tipe video, rasio kompresi yang didapat berbeda-beda karena bagian tubuh yang direkam dalam *frame-frame* video tersebut berbeda. Rasio kompresi paling besar diperoleh dari video 'MR chest' dengan nilai rasio kompresi mencapai angka 54 karena pada video tersebut banyak memiliki redundansi spasial seperti terlihat pada Gambar 4.16. Ketika mengalami kuantisasi nilai redundansi tersebut akan menyebabkan koefisien bernilai nol dalam jumlah banyak sehingga tidak dihitung dalam perkiraan rasio kompresi data *encode*. Ketika mengalami kuantisasi nilai redundansi tersebut akan menyebabkan koefisien bernilai nol dalam jumlah banyak sehingga tidak dihitung dalam prediksi rasio kompresi data *encode*.

Disamping itu, pada *frame* ke-1, 2, dan 3 tidak terjadi perubahan temporal sehingga tidak ada *motion vector* yang memiliki koefisien tidak bernilai nol seperti terlihat pada Gambar 4.17. Dengan demikian terjadi banyak pengurangan redundansi temporal dan nilai rasio kompresi menjadi sangat besar.

Gambar 4. 16 *Frame* video 'MR chest'



Gambar 4. 17 Video MR Chest dalam potongan *frame*

Nilai rasio kompresi dapat diubah-ubah atau disesuaikan dengan kebutuhan dengan mengubah-ubah parameter kuantisasi pada simulasi. Semakin besar nilai parameter kuantisasi maka semakin besar pula rasio kompresi namun nilai PSNR akan menurun. Dengan demikian, perlu dicari parameter kuantisasi optimum untuk melakukan kompresi.

Dari Gambar 4.8 hingga Gambar 4.13 dapat dilihat bahwa tren nilai rasio kompresi konsisten untuk hasil simulasi H.264 dengan DCT dan DWT pada setiap jenis video. Hal ini terlihat dari penurunan yang terjadi pada frame ke-1,5, dan 9 dimana dilakukan prediksi spasial.

BAB 5
KESIMPULAN

Rancangan simulasi kompresi video medis berdasarkan algoritma H.264 dengan menggunakan transformasi wavelet diskrit telah dibuat dan diuji. Simulasi kompresi berdasarkan algoritma H.264 baik dengan DCT maupun dengan DWT dilakukan dengan nilai parameter kompresi yang sama. Kualitas hasil simulasi kompresi juga sudah diuji dengan penilaian kualitas objektif, yakni parameter PSNR dan rasio kompresi. Dengan penggunaan parameter kompresi yang sama, didapatkan nilai PSNR DWT yang mengungguli nilai PSNR kompresi dengan DCT dengan *gain* hingga 1,06 dB pada video 'CT aperts' dengan filter Coiflet3. Rasio kompresi yang didapat dari simulasi menunjukkan bahwa secara *general* DWT memiliki rasio kompresi yang juga lebih unggul daripada DCT. Namun pada beberapa jenis filter DWT, rasio kompresinya lebih rendah disebabkan oleh *padding* yang dialokasikan oleh fungsi dekomposisi wavelet yang bertujuan untuk mengurangi distorsi tepi pada *sub-band* hasil dekomposisi.

DAFTAR REFERENSI

[1] Hanif Al Fatta. 2010. Telemedicine: dalam tinjauan teknologi informasi. Jurnal DASI Amikom.

[2] Yaslis Ilyas. 2006. Determinan Distribusi Dokter Spesialis di Kota/Kabupaten Indonesia. Jurnal Manajemen Pelayanan Kesehatan Volume 09 No.3.

[3] Hongtao Yu, et. al.. 2005. Applications and Improvement of H.264 in Medical Video Compression. IEEE Transactions on Circuits and Systems.

[4] Min-Jen Tsai, et. al.. 1994. Coronary angiogram video compression. Nuclear Science Symposium and Medical Imaging Conference IEEE Conference Record.

[5] David Gibson, et. al.. 2001. Angiogram video compression using a wavelet-based texture modeling approach. British Conf. Medical Image Understanding and Analysis.

[6] Zixiang. Xiong, et. al.. 1999. A comparative study of DCT- and Wavelet-based image coding. IEEE Transactions on Circuits and Systems for Video Technology.

[7] Dadang Gunawan. 2001. Microcalcification Detection using Wavelet Transform", Proceeding PACRIM 2001 Canada, pp. 694 – 697.

[8] Dadang Gunawan. 2010. Wavelet Transform as a Tool in Signal Processing: Its application in Medical Images. Keynote Paper on "4th Indonesia Japan Joint Scientific Symposium 2010 (IJJSS-2010).

[9] Dadang Gunawan. 2010. The Application of Wavelet Transform for an Image Compression. Keynote paper on 2nd Makassar International Conference on Electrical Engineering and Informatics (MICEEI).

[10] Iain Richardson E.G. 2003. H-264 and MPEG-4 Video Compression. John Wiley & Sons, Ltd.

[11] Rafael C. Gonzalez, Richard E. Woods. 2002. Digital Image Processing Second Edition. Prentice Hall.

[12] Niedered, Peter., Lee, T. Clive. 2010, et.al. Basic Engineering for Medics and Biologists, Chapter V: Medical Imaging. Amsterdam. IOS Press.

[13] Ultrasound Machine Picture Ultrasonography. http://www.virtualmedicalcentre.com/health-investigation/ultrasound-ultrasound-scanning-or-sonography-and-doppler/8. Diakses tanggal 26 Juni 2012.

[14] Margaret A. Lloyd, et.al. 2004. Mayo Celebrates a Half Century of Echo Advances. Mayo Clinic Cardiovascular Update Volume 2 Number 1 2004.

[15] Angiography or Artheriography. http://www.vims.ac.in/healthcare/hs-angiography.html. Diakses tanggal 26 Juni 2012.

[16] Andrew B. Watson. 1994. Image Compression Using the Discrete Cosine Transform. Mathematica Journal, 4(1), p. 81-88, NASA Ames Research Center.

[17] Paul S. Addison. 2005. Wavelet transforms and the ECG: a review. Physiological Measurement, Institute of Physics Publishing.

[18] David Salomon. 2004. Data Compression, The Complete Reference, Third Edition. New York. Springer-Verlag.

[19] Panchamkumar D.Shukla. 2003. Complex Wavelet Transforms and Their Applications. Dissertation at Signal Processing Division, University of Strathclyde.

[20] Stèphane G. Mallat. 1998. A Wavelet Tour of Signal Processing. Academic Press.

[21] Introduction to The Wavelet Families. http://www.mathworks.com/help/toolbox/wavelet/gs/f3-998390.html. Diakses 6 Juni 2012.

[22] An explanation of video compression techniques. 2009. White paper AXIS Communication.

[23] Gary. J. Sullivan, et. al.. 2004. The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions. Presented at the SPIE Conference on Applications of Digital Image Processing XXVII Special Session on Advances in the New Emerging Standard: H.264/AVC.

[24] Aroh Barjatya. 2004. Block Matching Algorithm for Motion Estimation. DIP 6620 Spring 2004 Final Project Paper, Blekinge Institute of Technology.

[25] Priyanka Singh, et.al. 2011. JPEG Image Compression based on Biorthogonal, Coiflets and Daubechies Wavelet Families. International Journal of Computer Applications (0975-8887) Vol.13 No.1.

[26] Recommendation ITU-T H.264 : Advanced video coding for generic audiovisual services. 2011. ITU Series H: Audiovisual and Multimedia Systems: Infrastructure of audiovisual services- Coding of moving video.

[27] Border Effects. http://www.mathworks.com/help/toolbox/wavelet/ug/f8-25097.html. Diakses 6 Juni 2012.

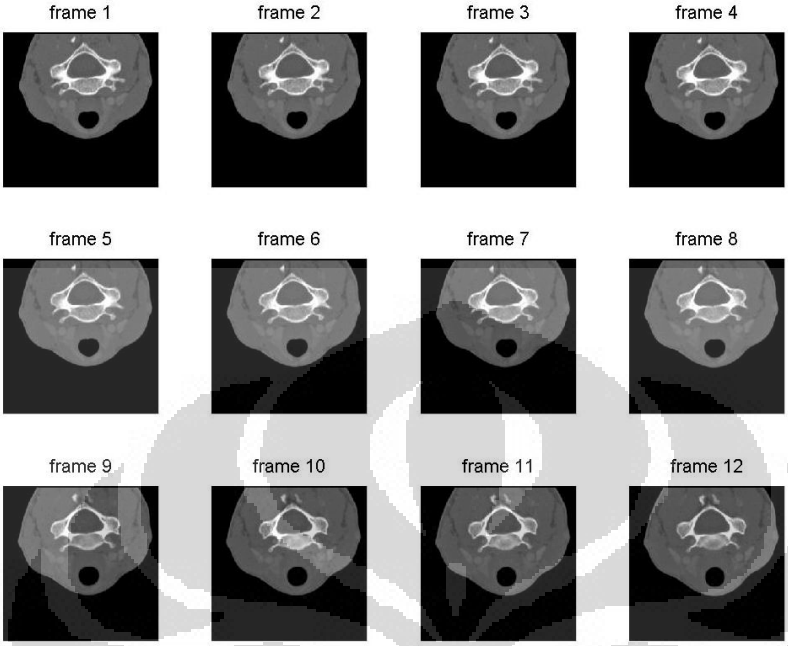
[28] Wavelet Browser by PyWavelets. 2008. http://wavelets.pybytes.com/. Diakses 2 Juni 2012.

[29] Satyabrata Rout. 2003. Orthogonal vs Biorthogonal Wavelets for Image Compression. Thesis at Virginia Polytechnic Institute and State University.

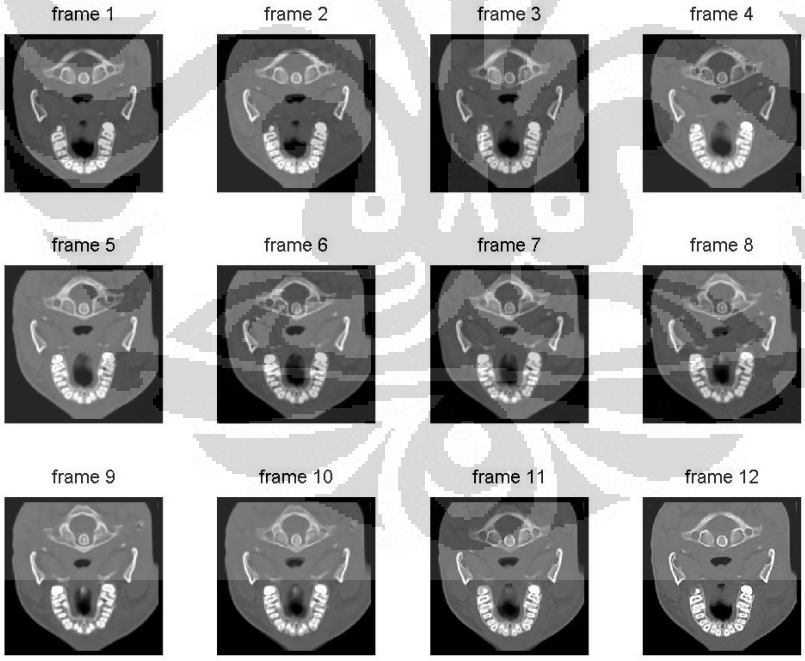
[30] Elios Klemo. 2005. Symlet and Gabor Wavelet Prediction of Printer Defects. Thesis at University of Kentucky.

Lampiran 1 : Potongan Frame Data Video Medis

CT Carotid

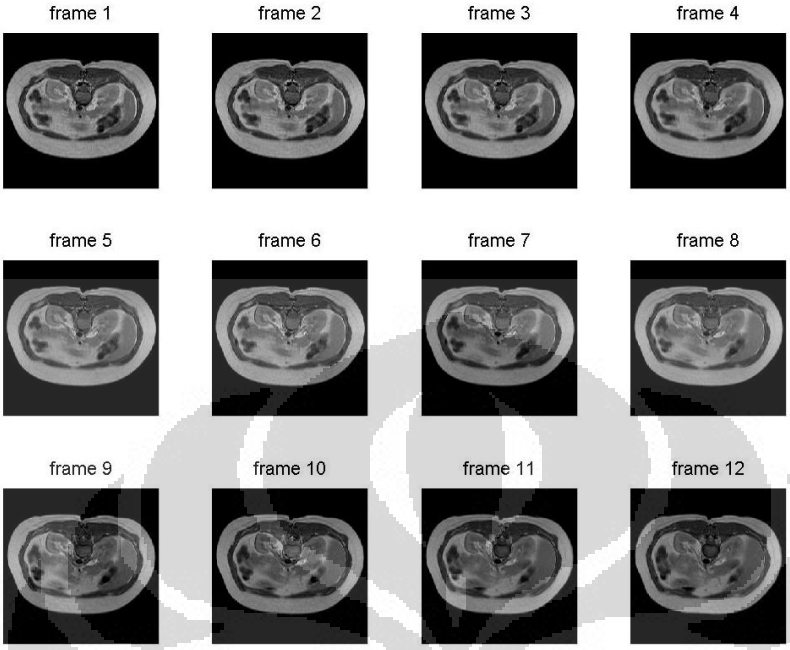


CT Skull

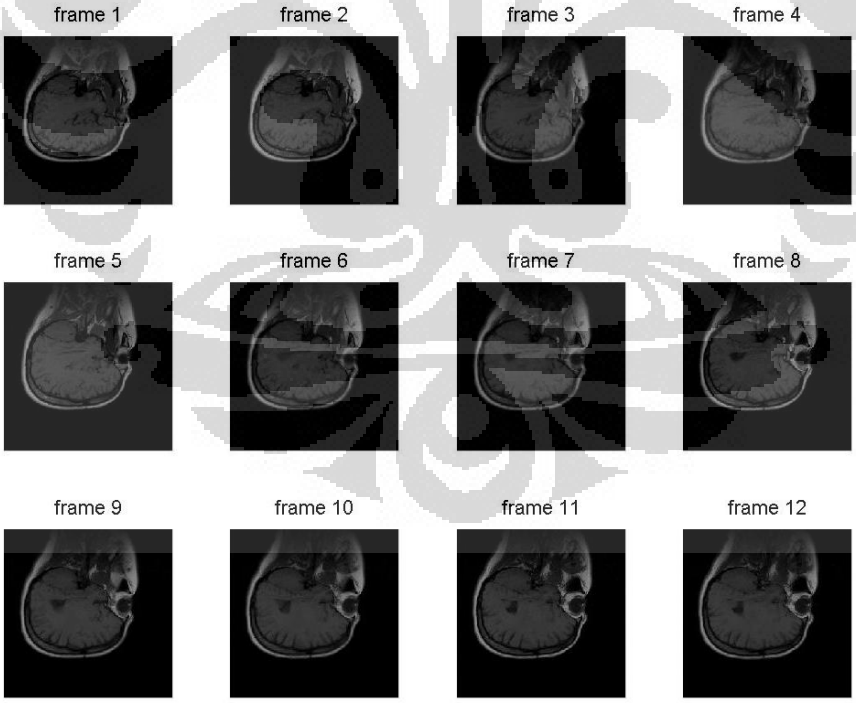


(lanjutan)

MR Liver

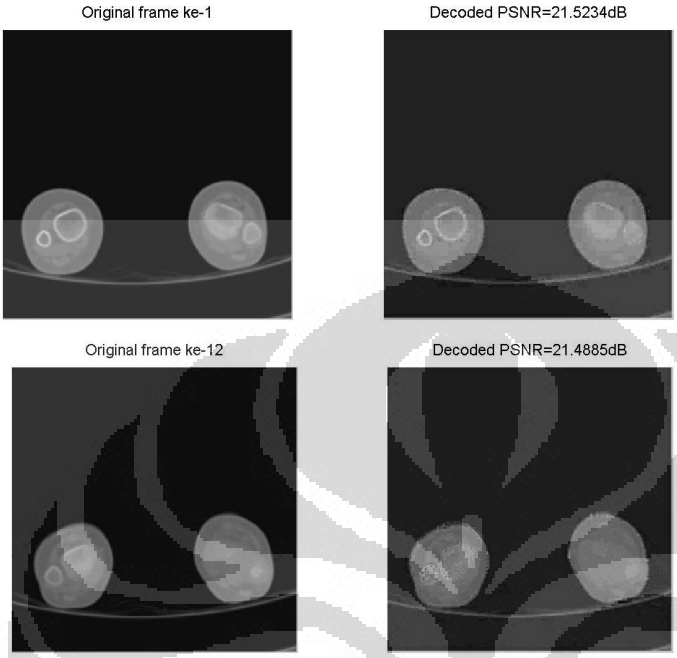


MR Head

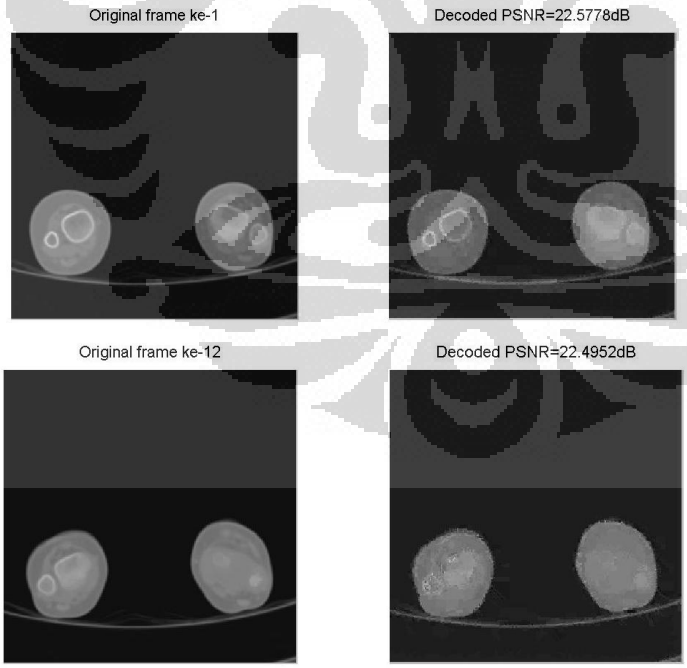


Lampiran 2 : Gambar Perbandingan Frame Orisinal vs Rekonstruksi DCT vs DWT

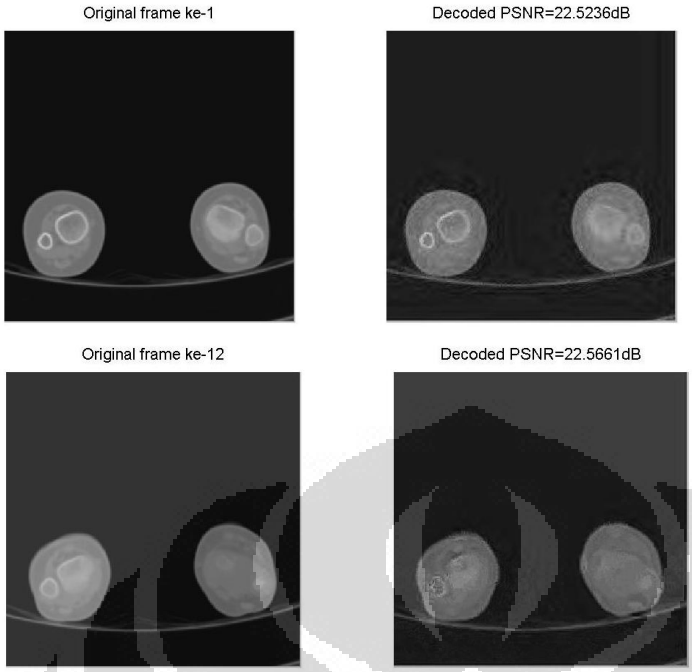
Video CT Aperts dengan Kompresi H.264 DCT (frame 1 dan frame 12):



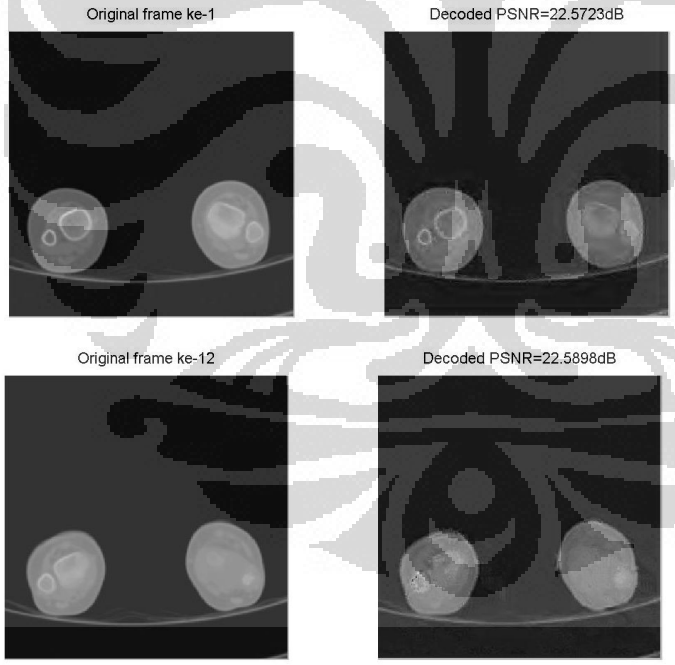
Video CT Aperts dengan Kompresi H.264 DWT ‘haar’ (frame 1 dan frame 12):



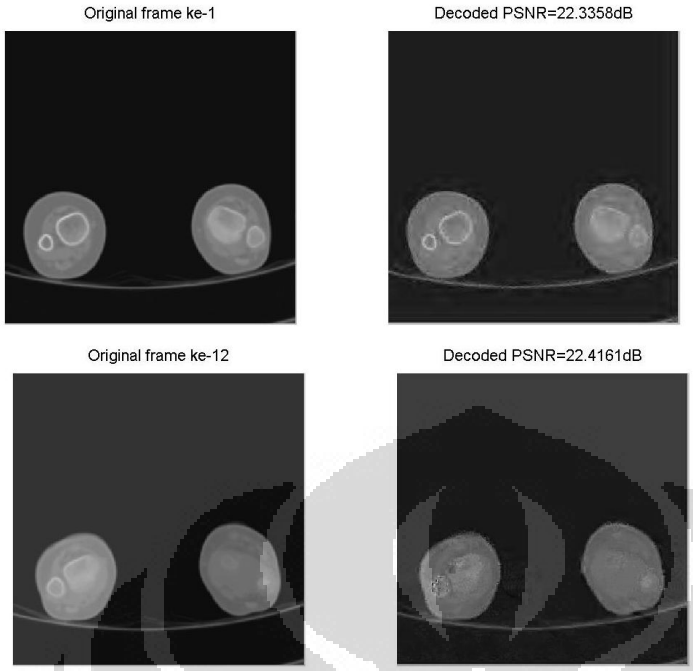
Video CT Aperts dengan Kompresi H.264 DWT ‘db8’ (frame 1 dan frame 12):



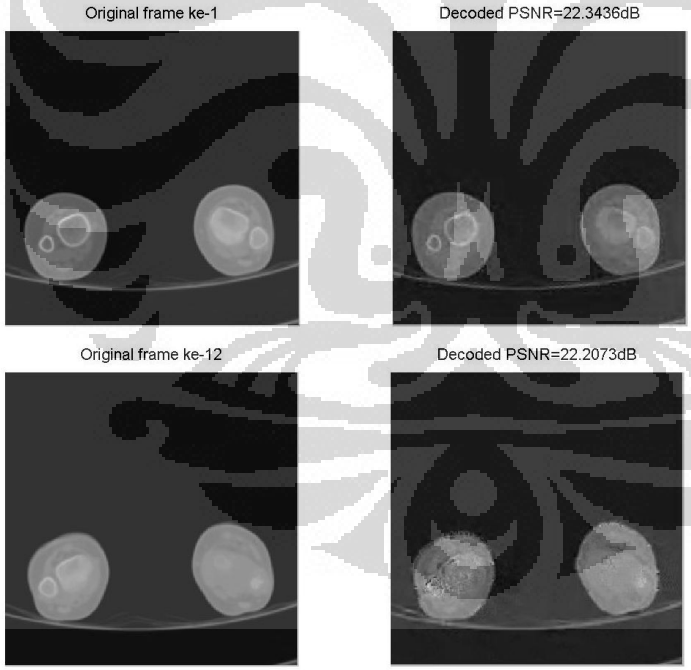
Video CT Aperts dengan Kompresi H.264 DWT ‘coif3’ (frame 1 dan frame 12):



Video CT Aperts dengan Kompresi H.264 DWT ‘sym4’ (frame 1 dan frame 12):

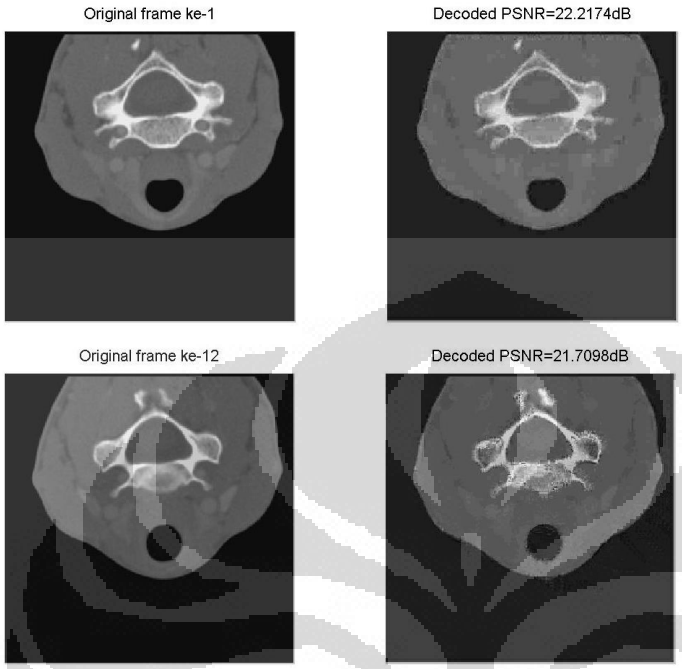


Video CT Aperts dengan Kompresi H.264 DWT ‘bior3.9’ (frame 1 dan frame 12):

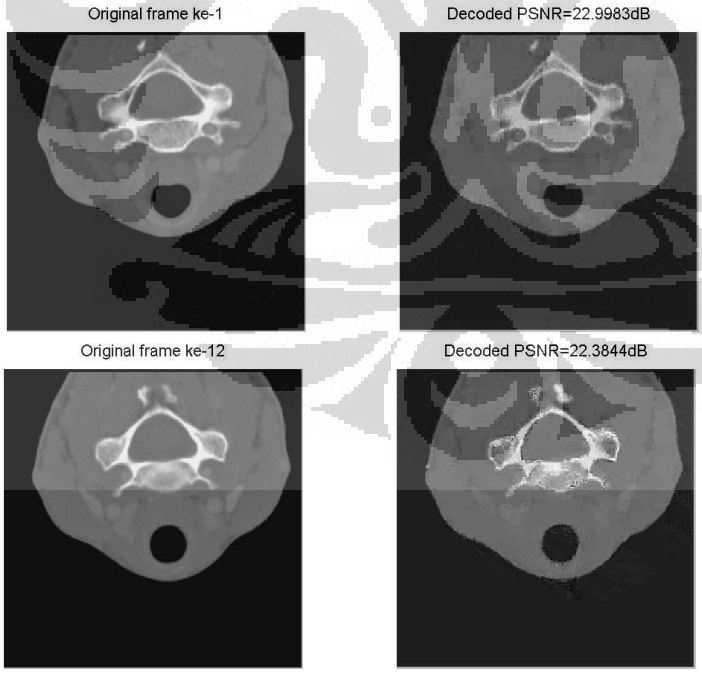


(lanjutan)

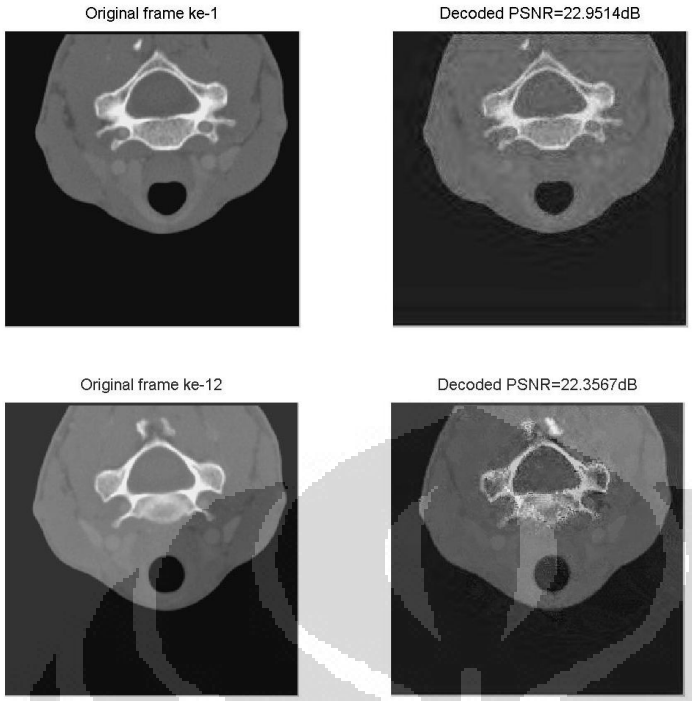
Video CT Carotid dengan Kompresi H.264 DCT (frame 1 dan frame 12):



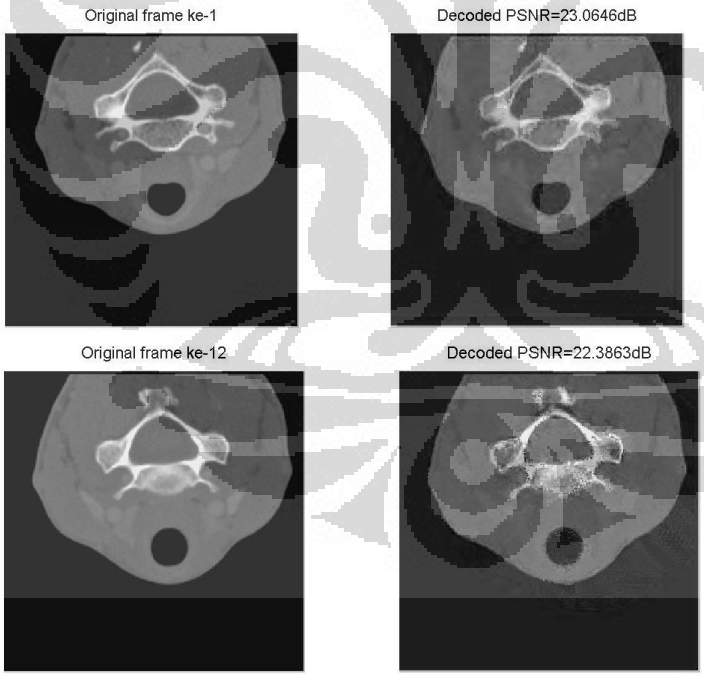
Video CT Carotid dengan Kompresi H.264 DWT ‘haar’ (frame 1 dan frame 12):



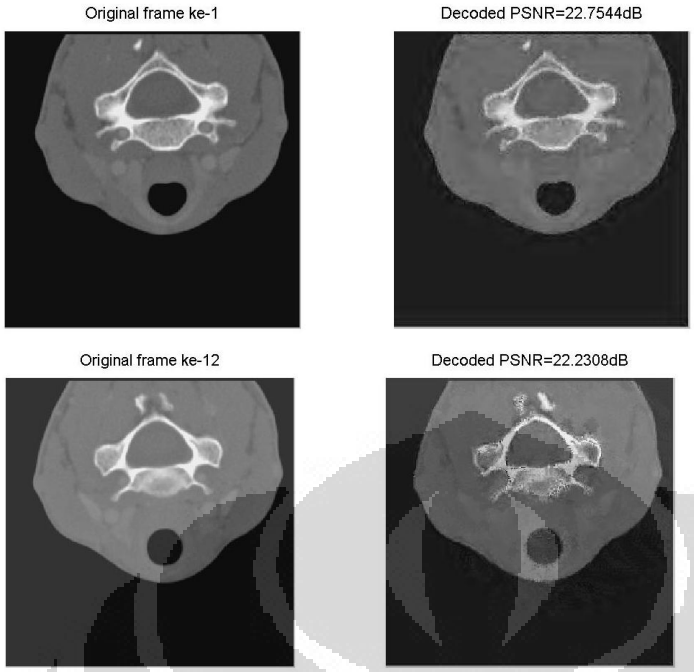
Video CT Carotid dengan Kompresi H.264 DWT ‘db8’ (frame 1 dan frame 12):



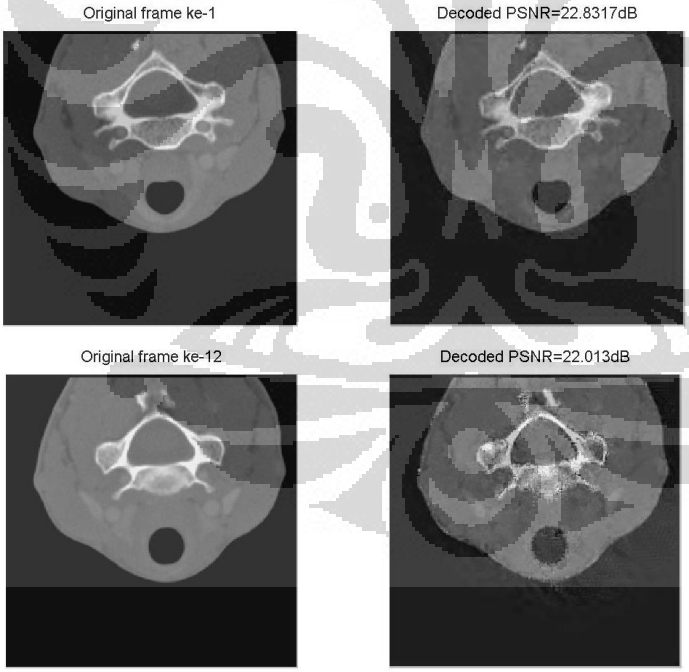
Video CT Carotid dengan Kompresi H.264 DWT ‘coif3’ (frame 1 dan frame 12):



Video CT Carotid dengan Kompresi H.264 DWT ‘sym4’ (frame 1 dan frame 12):

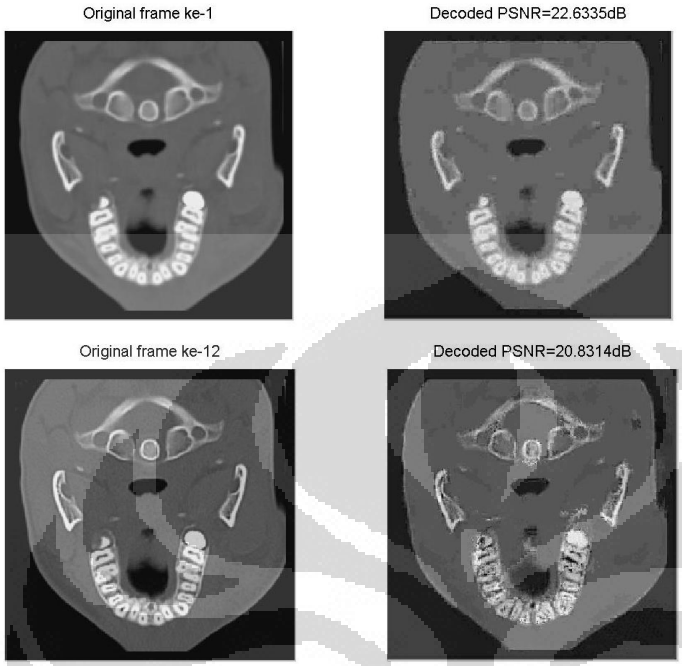


Video CT Carotid dengan Kompresi H.264 DWT ‘bior3.9’ (frame 1 dan frame 12):

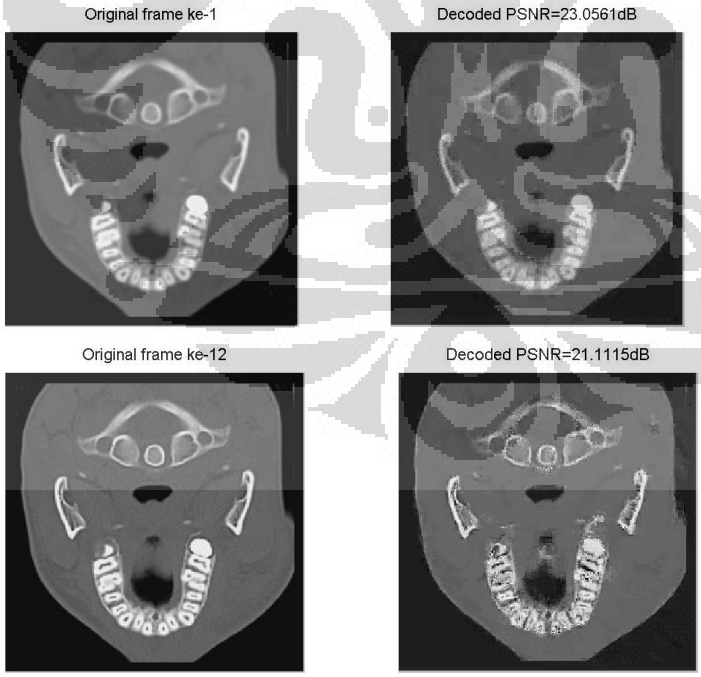


(lanjutan)

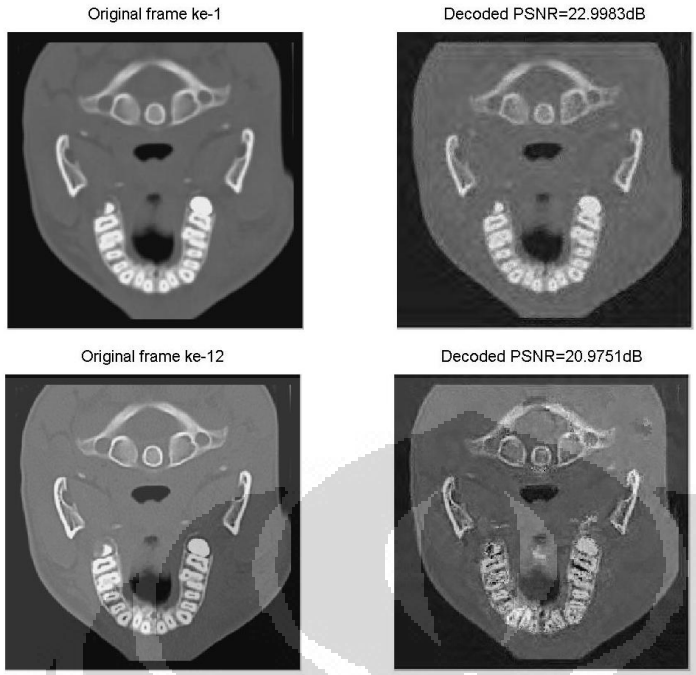
Video CT Skull dengan Kompresi H.264 DCT (frame 1 dan frame 12):



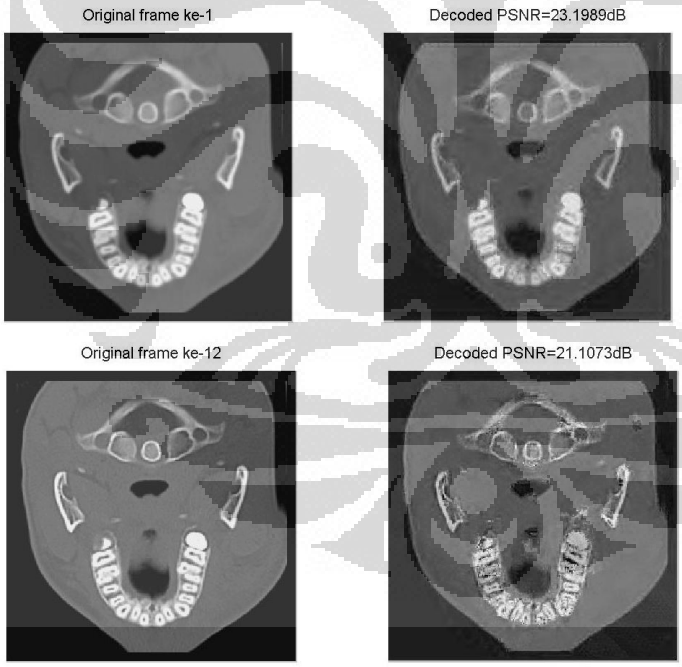
Video CT Skull dengan Kompresi H.264 DWT ‘haar’ (frame 1 dan frame 12):



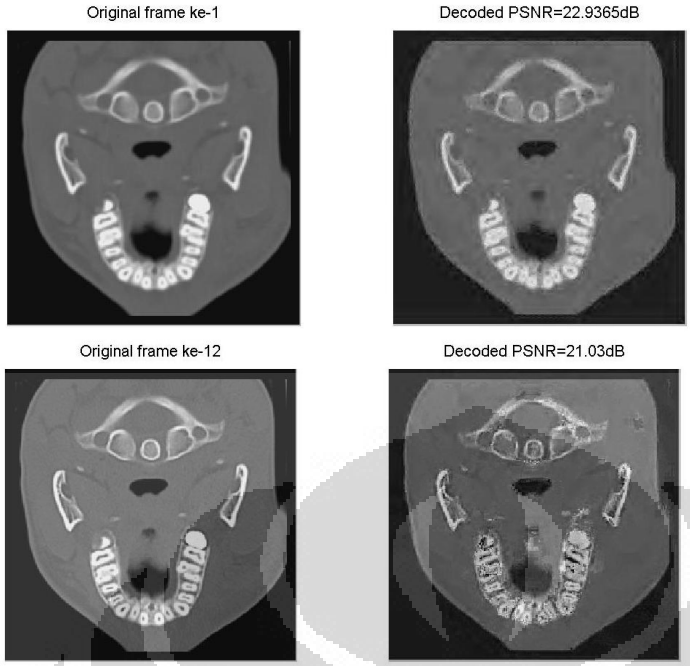
Video CT Skull dengan Kompresi H.264 DWT ‘db8’ (frame 1 dan frame 12):



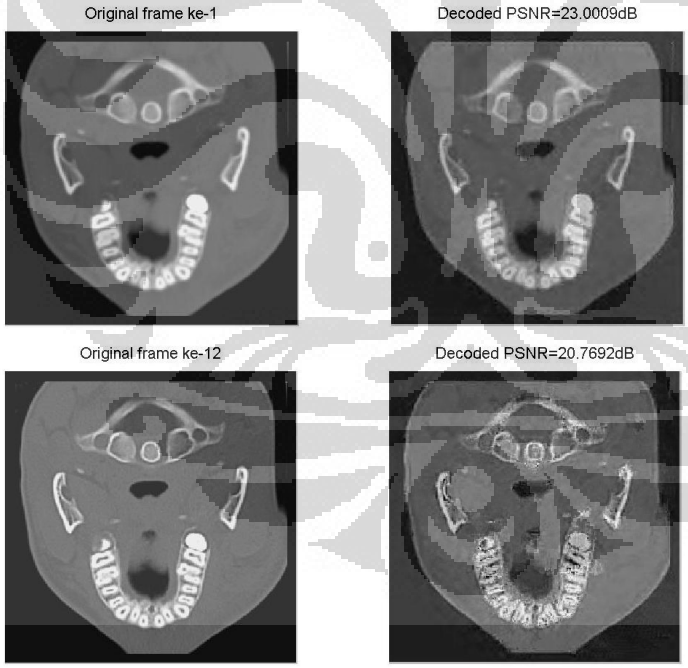
Video CT Skull dengan Kompresi H.264 DWT ‘coif3’ (frame 1 dan frame 12):



Video CT Skull dengan Kompresi H.264 DWT ‘sym4’ (frame 1 dan frame 12):

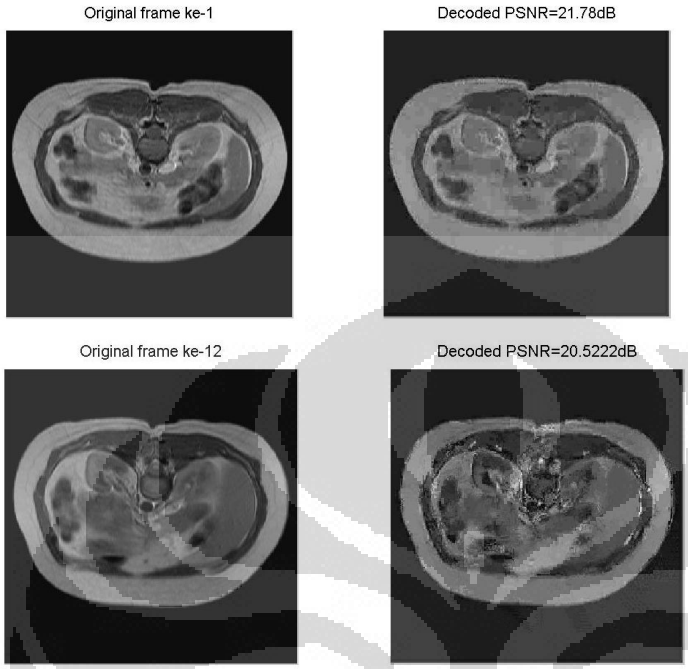


Video CT Skull dengan Kompresi H.264 DWT ‘bior3.9’ (frame 1 dan frame 12):

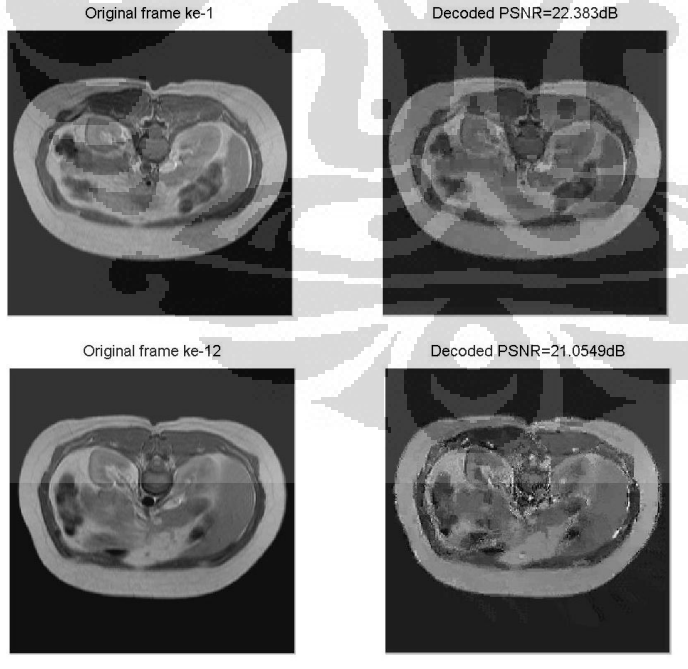


(lanjutan)

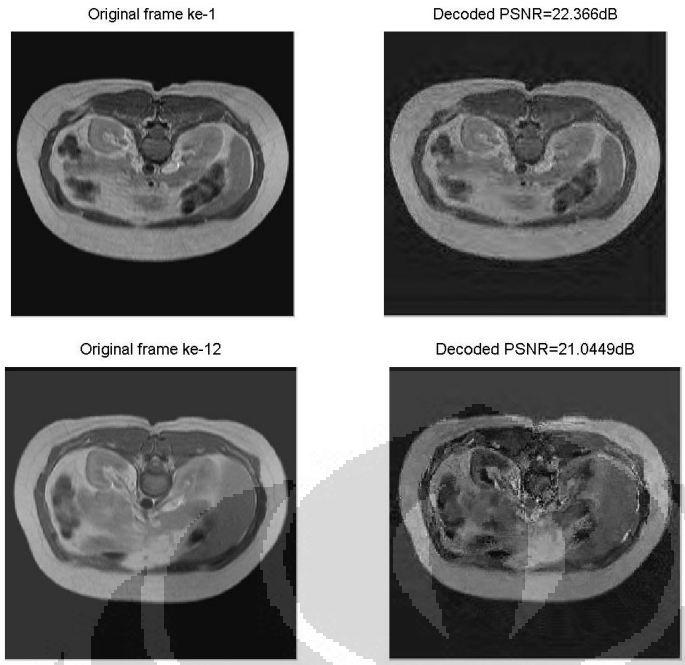
Video MR Liver dengan Kompresi H.264 DCT (frame 1 dan frame 12):



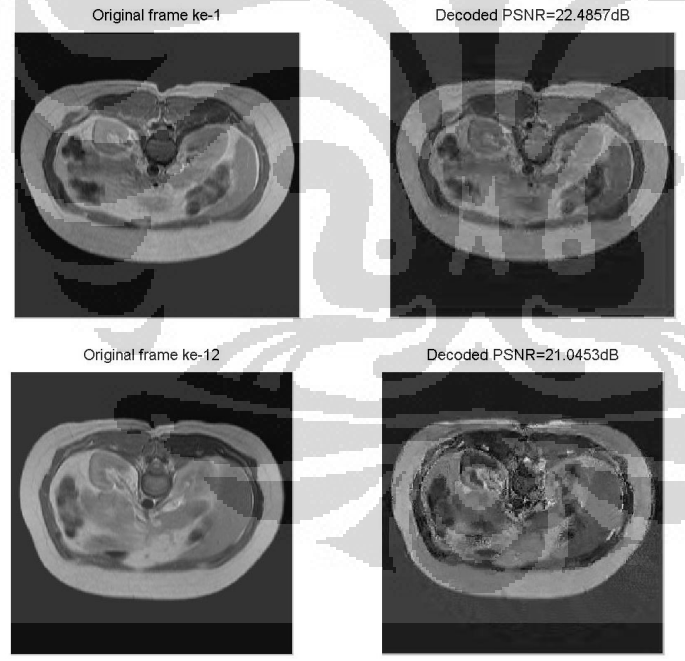
Video MR Liver dengan Kompresi H.264 DWT ‘haar’ (frame 1 dan frame 12):



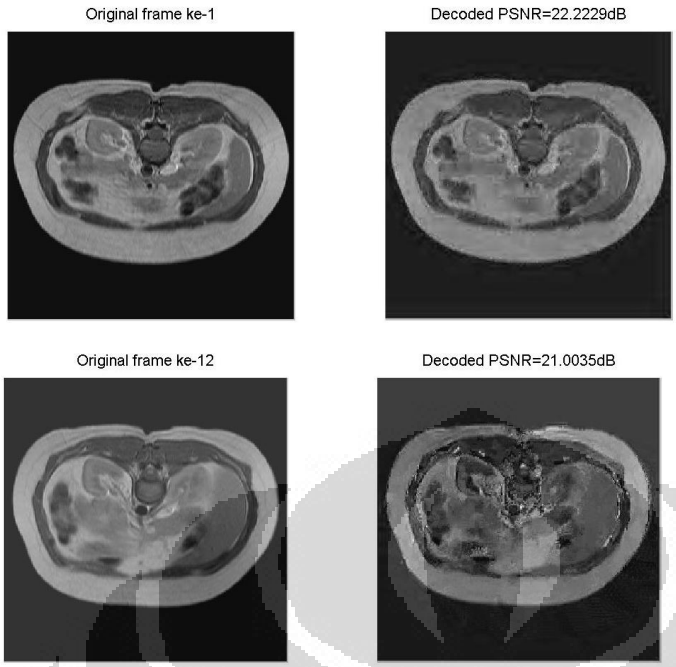
Video MR Liver dengan Kompresi H.264 DWT ‘db8’ (frame 1 dan frame 12):



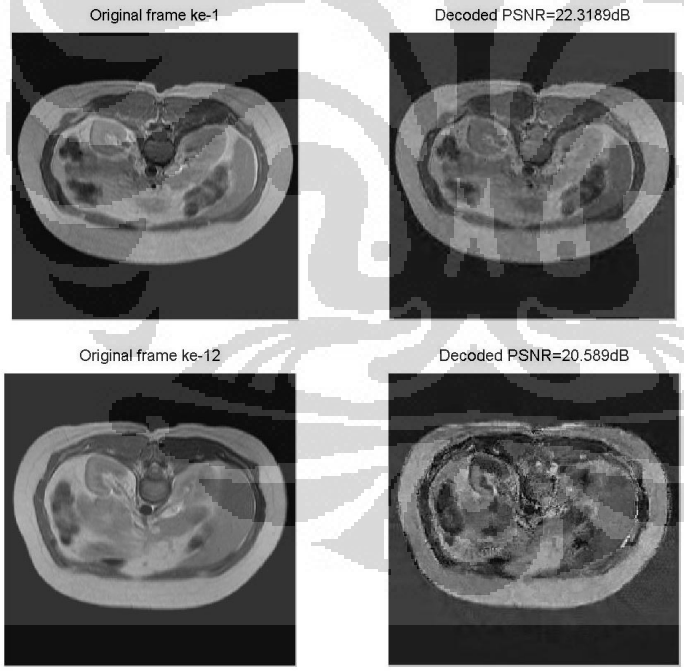
Video MR Liver dengan Kompresi H.264 DWT ‘coif3’ (frame 1 dan frame 12):



Video MR Liver dengan Kompresi H.264 DWT ‘sym4’ (frame 1 dan frame 12):

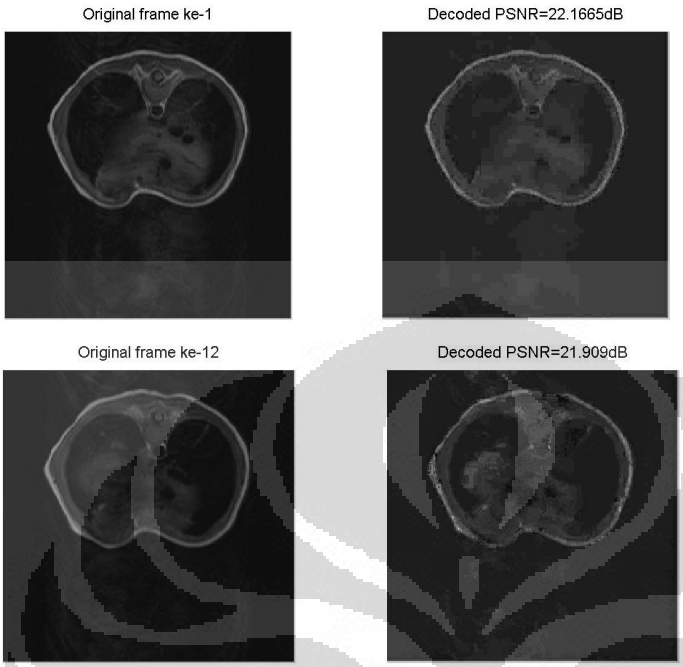


Video MR Liver dengan Kompresi H.264 DWT ‘bior3.9’ (frame 1 dan frame 12):

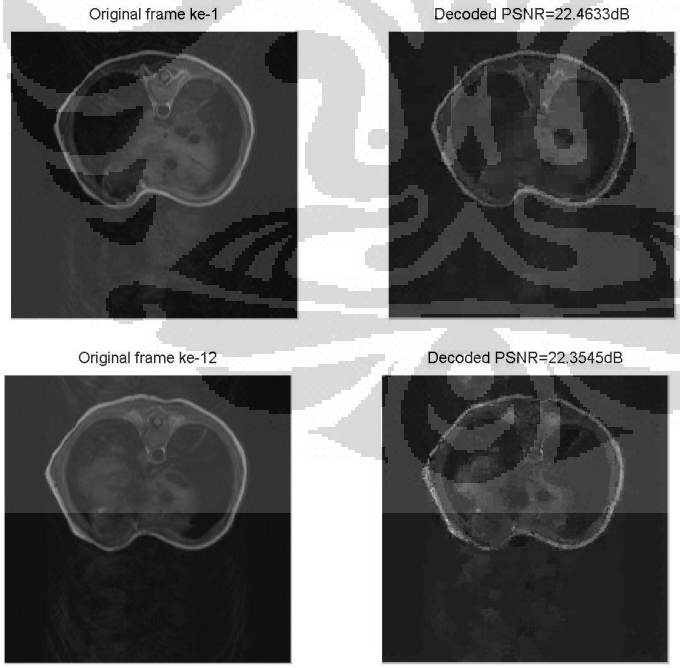


(lanjutan)

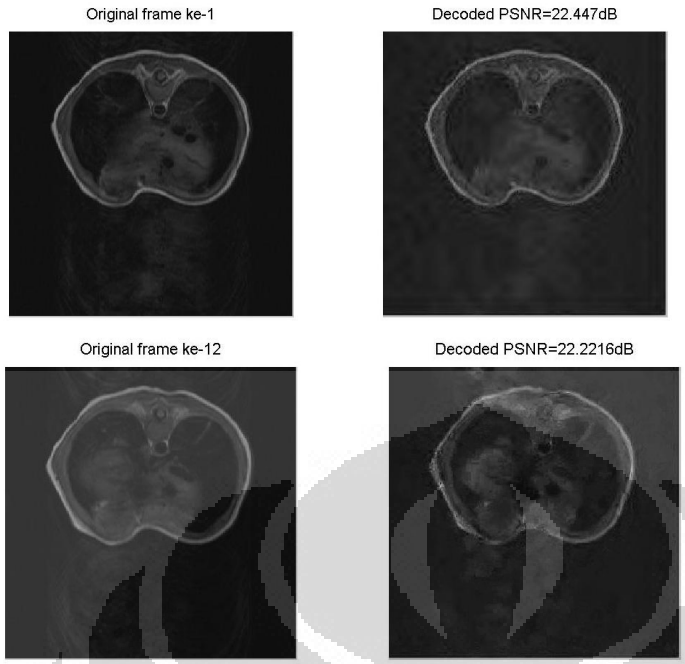
Video MR Chest dengan Kompresi H.264 DCT (frame 1 dan frame 12):



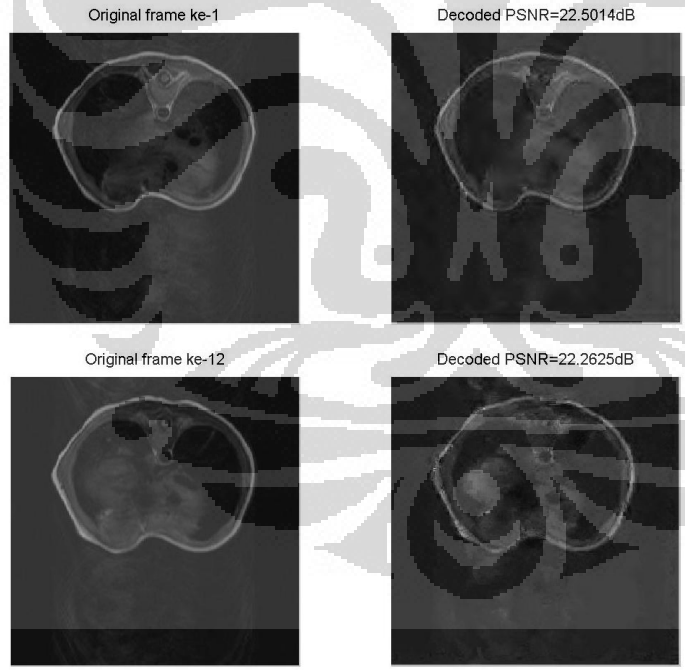
Video MR Chest dengan Kompresi H.264 DWT ‘haar’ (frame 1 dan frame 12):



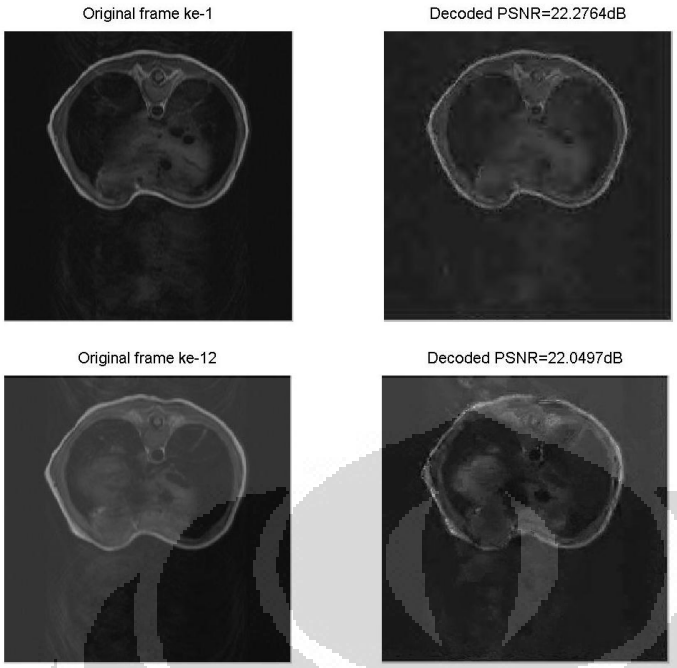
Video MR Chest dengan Kompresi H.264 DWT ‘db8’ (frame 1 dan frame 12):



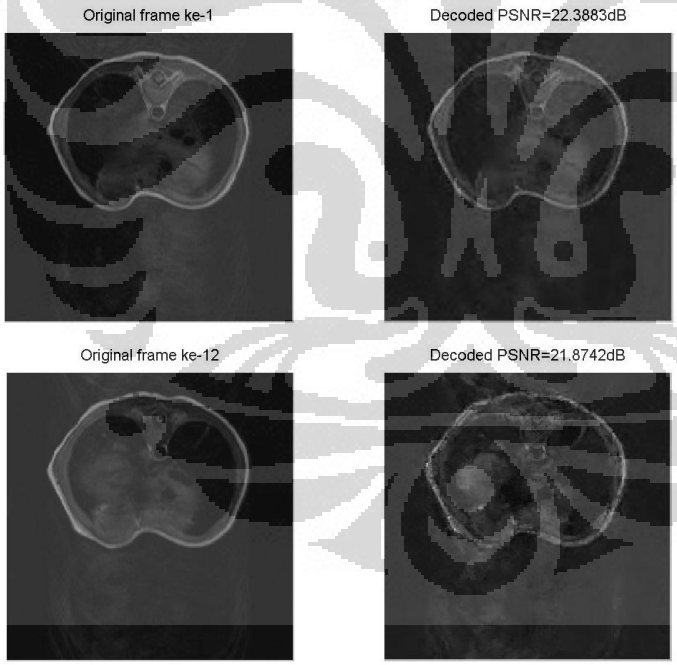
Video MR Chest dengan Kompresi H.264 DWT ‘coif3’ (frame 1 dan frame 12):



Video MR Chest dengan Kompresi H.264 DWT ‘sym4’ (frame 1 dan frame 12):

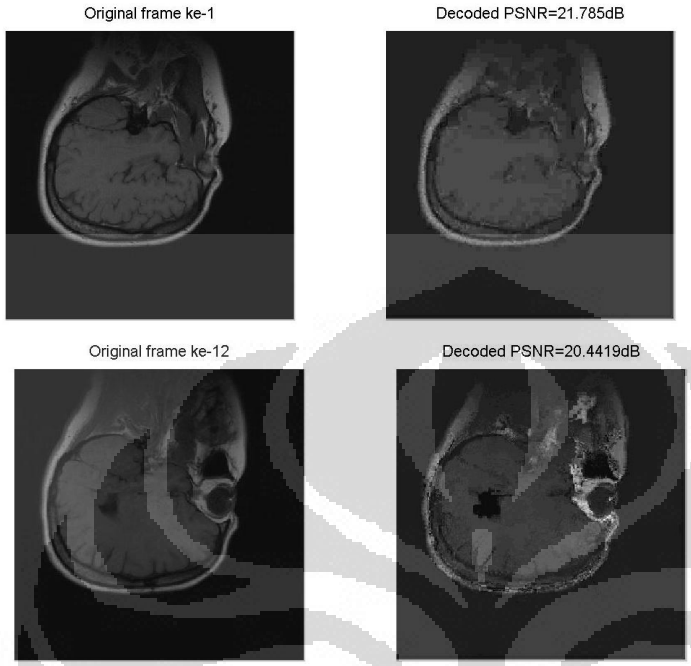


Video MR Chest dengan Kompresi H.264 DWT ‘bior3.9’ (frame 1 dan frame 12):

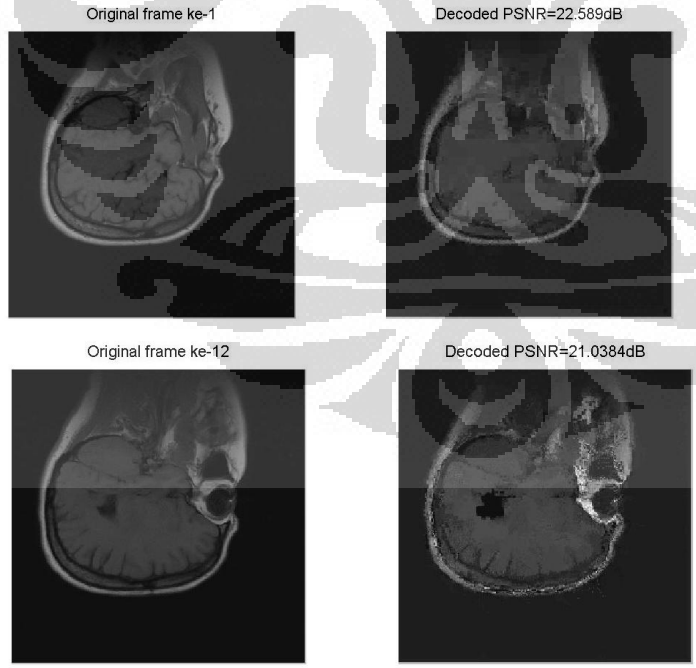


(lanjutan)

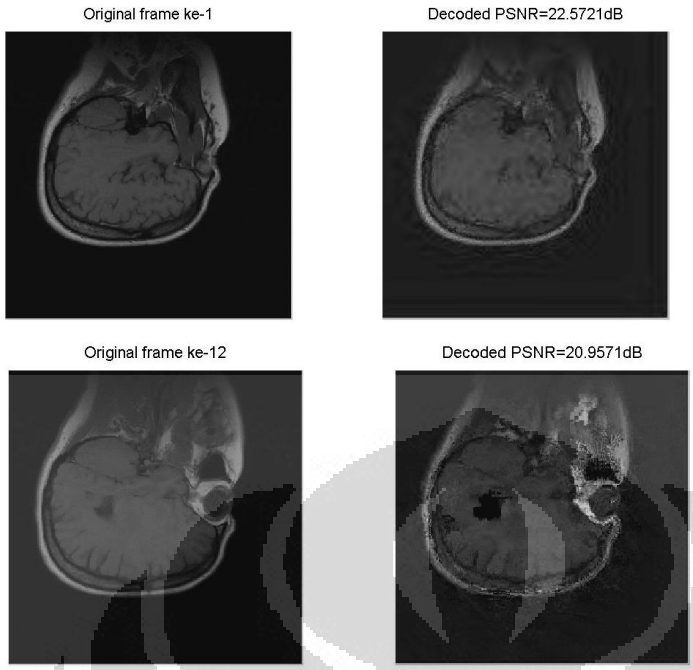
Video MR Head dengan Kompresi H.264 DCT (frame 1 dan frame 12):



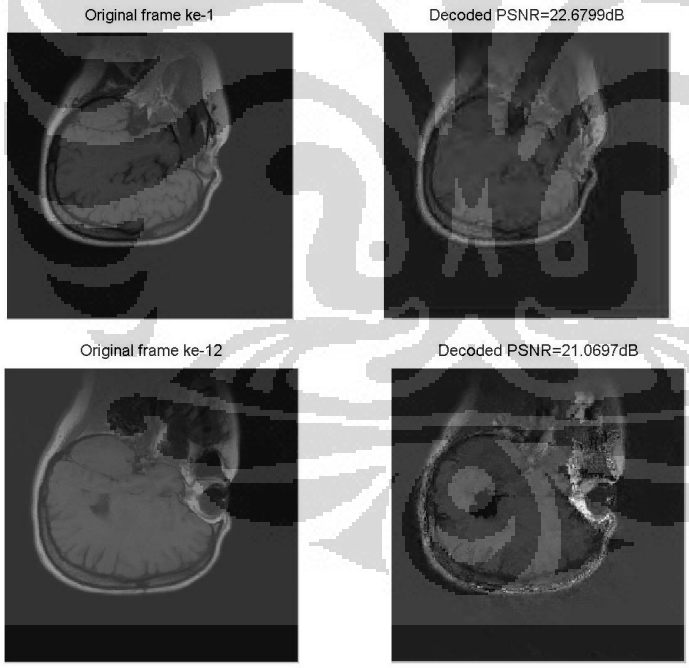
Video MR Head dengan Kompresi H.264 DWT ‘haar’ (frame 1 dan frame 12):



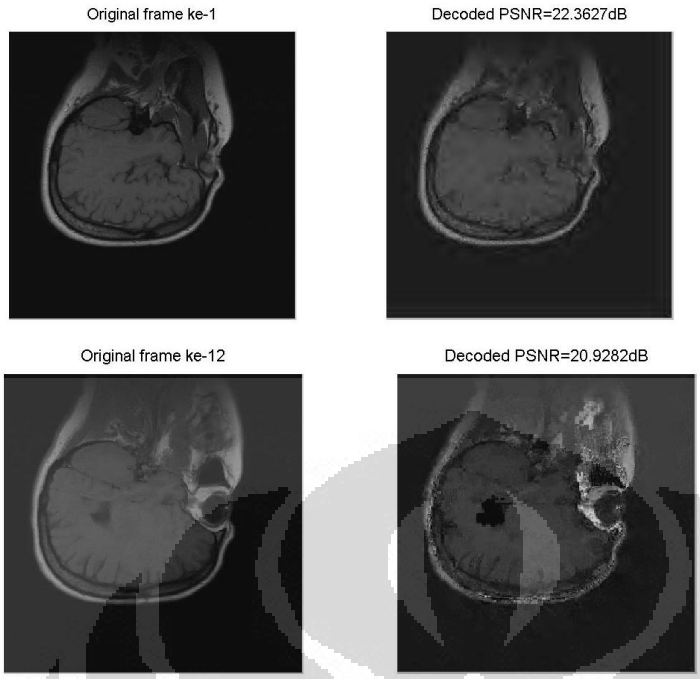
Video MR Head dengan Kompresi H.264 DWT ‘db8’ (frame 1 dan frame 12):



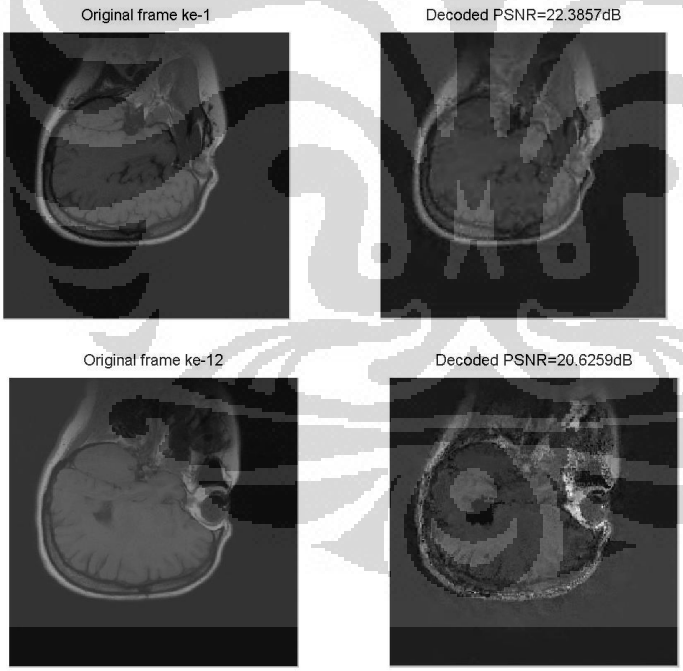
Video MR Head dengan Kompresi H.264 DWT ‘coif3’ (frame 1 dan frame 12):



Video MR Head dengan Kompresi H.264 DWT ‘sym4’ (frame 1 dan frame 12):



Video MR Head dengan Kompresi H.264 DWT ‘bior3.9’ (frame 1 dan frame 12):



- Algoritma H.264 dengan DCT

```
%Baseline profile H.264 algorithm Using DCT
%Annisa Dinda Amalia

%ENCODER
tic;
clear;close all;

%baca video sebagai frame
vidRaw=mmreader('filename1-12.avi');
%hitung jumlah frame
nFrames=vidRaw.NumberOfFrames;

%frame referensi pertama untuk frame P
previous=read(vidRaw,1);
previous=rgb2ycbcr(previous);
previous=double(imresize(previous(:,:,1),[256,256]));

%frameI processing
for(i=1:nFrames)
    if i==1 || i==5 || i==9
        %baca frame ke-i, ubah ke ycbcr, ambil y
        %ubah ke double dan resize
        frameI=read(vidRaw,i);
        frameIntra=rgb2ycbcr(frameI);
        frameIntra=double(imresize(frameIntra(:,:,1),[256 256]));
        %intra prediction
        predictIntra=blkproc(frameIntra,[5 5],@modeDC);
        residuIntra=frameIntra-predictIntra;
        dctIntra=blkproc(residuIntra,[4 4],@dct264); %forward DCT residu
        quantIntra=round(dctIntra*0.025); %kuantisasi QP=36
        scanIntra=zigzag(quantIntra); %reordering hasil kuantisasi residu
        filename=sprintf('%s%d.mat','Intra',i);
        save(filename,'scanIntra');  %save hasil zigzag reorder

        dctpredictIntra=blkproc(predictIntra,[4 4],@dct264);
        quantpredictIntra=round(dctpredictIntra*0.025);
        scanpredictIntra=zigzag(quantpredictIntra);
        filename=sprintf('%s%d.mat','predictIntra',i);
        save(filename,'scanpredictIntra');

    else

        %frame P processing
        %baca frame ke-i, ubah ke ycbcr, ambil y
        %ubah ke double dan resize
        frameP=read(vidRaw,i);
        frameInter=rgb2ycbcr(frameP);
        frameInter=double(imresize(frameInter(:,:,1),[256 256]));
        %inisialisasi ukuran makroblok dan daerah pencarian
        mbSize=4;p=3;
        %menghitung motion vector dengan TSS
        motionVector=motionEstTSS(frameInter,previous,mbSize,p);
        [motrow motcol]=size(motionVector);
        %menghitung citra kompensasi dari motion vector yg didapat
        imgCompensated=motionComp(previous,motionVector,mbSize);
        residuInter=frameInter-imgCompensated;
        dctInter=blkproc(residuInter,[4 4],@dct264); %fwd DCT dari residu
        quantInter=round(dctInter*0.025); %kuantisasi dg QP=36
```

```
        previous=imgCompensated; %referensi berikutnya dari rekonstruksi

        scanInter=zigzag(quantInter);  %reordering
        filename=sprintf('%s%d.mat','Inter',i);
        save(filename,'scanInter');   %simpan hasil reordering

        scanmotion=zigzag(motionVector);
        filename=sprintf('%s%d.mat','motion',i);
        save(filename,'scanmotion');

    end
end
%END of ENCODER

%HITUNG BIT DATA
sizeframe=zeros(1,12);

%data residu intraframe dan interframe
for i=1:12

intracount=0;
intercount=0;
predictintracount=0;
motioncount=0;

    if i==1 || i==5 || i==9
    filename=sprintf('%s%d.mat','Intra',i);
    data=load(filename);
    data=data.scanIntra;
    [row col]=size(data);
    for k=1:col
        if data(1,k)==0
            intracount=intracount;
        else
            intracount=intracount+1;
        end
    end

    filename=sprintf('%s%d.mat','predictIntra',i);
    data=load(filename);
    data=data.scanpredictIntra;
    [row col]=size(data);
    for k=1:col
        if data(1,k)==0
            predictintracount=predictintracount;
        else
            predictintracount=predictintracount+1;
        end
    end

    dataintra=intracount+predictintracount;
    sizeframe(i)=dataintra*8;

    else

    filename=sprintf('%s%d.mat','Inter',i);
    data=load(filename);
    data=data.scanInter;
    [row col]=size(data);
    for k=1:col
        if data(1,k)==0
            intercount=intercount;
        else
            intercount=intercount+1;
        end
```

```
    end

    filename=sprintf('%s%d.mat','motion',i);
    data=load(filename);
    data=data.scanmotion;
    [row col]=size(data);
        for k=1:col
            if data(1,k)==0
            motioncount=motioncount;
            else
            motioncount=motioncount+1;
            end
        end

    datainter=intercount+motioncount;
    sizeframe(i)=datainter*8;

    end
end
save('sizeframe','sizeframe');

total_data=0;
for i=1:12
    total_data=total_data+sizeframe(i);
end

save ('total_size_dct','total_data');
%END of BIT DATA

%DECODER
aviobj=avifile('decodedctfilename1-
12.avi','compression','none','fps',30);

%load prediction Intra frame 1
        predictIntra=load ('predictIntra1');
        predictIntra=predictIntra.scanpredictIntra;
        [row col]=size(predictIntra);
        n=sqrt(col);
        predictIntra=invzigzag(predictIntra,n,n);
        dequantpredictIntra=predictIntra*40;
        invdctpredictIntra=blkproc(dequantpredictIntra,[4 4],@idct264);
        predictIntra=invdctpredictIntra;
        %load data residu frame 1 yg di-code
        dataIntra=load ('Intra1');
        dataIntra=dataIntra.scanIntra;
        [row col]=size(dataIntra);
        n=sqrt(col);
        dataIntra=invzigzag(dataIntra,n,n); %reordering jadi matrix
        dequantIntra=dataIntra*40; %rescaling dengan QP=36
        invdctIntra=blkproc(dequantIntra,[4 4],@idct264); %inv DCT
        recframe=invdctIntra+predictIntra; %rekonstruksi dengan
prediction Intra
        figure,imshow(uint8(recframe));
        toVid=getframe;
        aviobj=addframe(aviobj,toVid);

%load motion vector 2 untuk referensi awal frame P
motion=load('motion2');
motion=motion.scanmotion;
motion=invzigzag(motion,motrow,motcol);
imgComp=motionComp(recframe,motion,4);

for(i=2:12)
    %decode frame I
    if i==5 || i==9
```

```
        %load prediction Intra frame
        filename=sprintf('%s%d.mat','predictIntra',i);
        predictIntra=load (filename);
        predictIntra=predictIntra.scanpredictIntra;
        [row col]=size(predictIntra);
        n=sqrt(col);
        predictIntra=invzigzag(predictIntra,n,n);
        dequantpredictIntra=predictIntra*40;
        invdctpredictIntra=blkproc(dequantpredictIntra,[4 4],@idct264);
        predictIntra=invdctpredictIntra;
        %load data residu setiap frame I yg di-code
        filename=sprintf('%s%d.mat','Intra',i);
        dataIntra=load (filename);
        dataIntra=dataIntra.scanIntra;
        [row col]=size(dataIntra);
        n=sqrt(col);
        dataIntra=invzigzag(dataIntra,n,n); %reordering jadi matrix
        dequantIntra=dataIntra*40; %rescaling dengan QP=36
        invdctIntra=blkproc(dequantIntra,[4 4],@idct264); %inv DCT
        recframe=invdctIntra+predictIntra; %rekonstruksi dengan
prediction Intra
        figure,imshow(uint8(recframe));
        toVid=getframe;
        aviobj=addframe(aviobj,toVid);

    %decode frame P
    else
        filename=sprintf('%s%d.mat','Inter',i);
        dataInter=load (filename);
        dataInter=dataInter.scanInter;
        [row col]=size(dataInter);
        n=sqrt(col);
        dataInter=invzigzag(dataInter,n,n);
        dequantInter=dataInter*40;
        invdctInter=blkproc(dequantInter,[4 4],@idct264);
        recframe=invdctInter+imgComp;

        figure,imshow(uint8(recframe));
        toVid=getframe;
        aviobj=addframe(aviobj,toVid);
    end

    if i<12 && i~=4 && i~=8
            filename=sprintf('%s%d.mat','motion',i+1);
            motionVector=load(filename);
            motionVector=motionVector.scanmotion;
            motionVector=invzigzag(motionVector,motrow,motcol);

            mbSize=4;
            imgComp=motionComp(recframe,motionVector,mbSize);
    end

end

aviobj=close(aviobj);
%END of DECODER

%HITUNG COMP RATIO
%size original video
info=aviinfo('filename1-12.avi');
size=info.FileSize;
sizebit=size*8;

%rasio kompresi DCT
size_dct=load('total_size_dct');
```

```
size_dct=size_dct.total_data;
compratio_dct=sizebit/size_dct

%rasio kompresi tiap frame
sizeframe=load('sizeframe');
sizeframe=sizeframe.sizeframe;
compframe=zeros(1,12);
for i=1:12
    compframe(i)=(sizebit/12)/sizeframe(i);
end
save('compframe','compframe');
figure,plot(compframe);
saveas(gcf,'CRdct filename.fig');
%END of COMP RATIO

%HITUNG PSNR
vidOriginal=mmreader('filename1-12.avi');
nFrame=vidOriginal.NumberOfFrames;

vidDecoded=mmreader('decodedctfilename1-12.avi');
nFrames=vidDecoded.NumberOfFrames;

psnr=zeros([1 nFrame]);
for i=1:nFrame
    frameOri=read(vidOriginal,i);
    frameOri=rgb2ycbcr(frameOri);
    frameOri=double(frameOri(:,:,1));

    frameDecode=read(vidDecoded,i);
    frameDecode=rgb2ycbcr(frameDecode);
    frameDecode=double(frameDecode(:,:,1));

    psnr(i)=imgPSNR(frameOri,frameDecode,255);
    figure,subplot(1,2,1),imshow(uint8(frameOri)),title(['Original frame
ke-' ,num2str(i)]);
    subplot(1,2,2),imshow(uint8(frameDecode)),title(['Decoded
PSNR=',num2str(psnr(i)),'dB']);
    name=sprintf('%s%d.jpg','PSNRdct filename',i);
    saveas(gcf,name);
end
save ('psnrDCT','psnr');
figure,plot(psnr);
saveas(gcf,'PSNRdct filename.fig');
%END of HITUNG PSNR

toc;
```

- Algoritma H.264 dengan DWT

```
%Baseline profile H.264 algorithm Using DWT
%Annisa Dinda Amalia

%ENCODER
tic;
clear;close all;

%baca video sebagai frame
vidRaw=mmreader('filename1-12.avi');
%hitung jumlah frame
nFrames=vidRaw.NumberOfFrames;

%frame referensi pertama utk frame P
```

```
previous=read(vidRaw,1);
previous=rgb2ycbcr(previous);
previous=double(imresize(previous(:,:,1),[256,256]));

%frameI processing
for(i=1:nFrames)
    if i==1 || i==5 || i==9
    %baca frame ke-i, ubah ke ycbcr, ambil y
    %ubah ke double dan resize
    frameI=read(vidRaw,i);
    frameIntra=rgb2ycbcr(frameI);
    frameIntra=double(imresize(frameIntra(:,:,1),[256 256]));
    %intra prediction
        predictIntra=blkproc(frameIntra,[5 5],@modeDC);
        residuIntra=frameIntra-predictIntra;
        [dwtIntra sizes]=wavedec2(residuIntra,4,'coif3'); %forward dwt
residu
        quantIntra=round(dwtIntra*0.025); %kuantisasi QP=36
        scanIntra=zigzag(quantIntra); %reordering hasil kuantisasi residu
        filename=sprintf('%s%d.mat','IntraDWT',i);
        save(filename,'scanIntra');  %save hasil zigzag reorder

        dwtpredictIntra=wavedec2(predictIntra,4,'coif3');
        quantpredictIntra=round(dwtpredictIntra*0.025);
        scanpredictIntra=zigzag(quantpredictIntra);
        filename=sprintf('%s%d.mat','predictIntraDWT',i);
        save(filename,'scanpredictIntra');

    else

%frame P processing
    %baca frame ke-i, ubah ke ycbcr, ambil y
    %ubah ke double dan resize
    frameP=read(vidRaw,i);
    frameInter=rgb2ycbcr(frameP);
    frameInter=double(imresize(frameInter(:,:,1),[256 256]));
        %inisialisasi ukuran makroblok dan daerah pencarian
        mbSize=4;p=3;
        %menghitung motion vector dengan TSS
        motionVector=motionEstTSS(frameInter,previous,mbSize,p);
        %menghitung citra kompensasi dari motion vector yg didapat
        imgCompensated=motionComp(previous,motionVector,mbSize);
        residuInter=frameInter-imgCompensated;
        dwtInter=wavedec2(residuInter,4,'coif3'); %fwd dwt dari residu
        quantInter=round(dwtInter*0.025); %kuantisasi dg QP=36
        previous=imgCompensated; %referensi berikutnya dari rekonstruksi

        scanInter=zigzag(quantInter);  %reordering
        filename=sprintf('%s%d.mat','InterDWT',i);
        save(filename,'scanInter');   %simpan hasil reordering

        scanmotion=zigzag(motionVector);
        filename=sprintf('%s%d.mat','motiondwt',i);
        save(filename,'scanmotion');


    end
end
%END of ENCODER


%HITUNG BIT DATA
sizeframe=zeros(1,12);

%data residu intraframe dan interframe
```

```
for i=1:12

intracount=0;
intercount=0;
predictintracount=0;
motioncount=0;

    if i==1 || i==5 || i==9
    filename=sprintf('%s%d.mat','IntraDWT',i);
    data=load(filename);
    data=data.scanIntra;
    [row col]=size(data);
    for k=1:col
        if data(1,k)==0
            intracount=intracount;
        else
            intracount=intracount+1;
        end
    end

    filename=sprintf('%s%d.mat','predictIntraDWT',i);
    data=load(filename);
    data=data.scanpredictIntra;
    [row col]=size(data);
    for k=1:col
        if data(1,k)==0
            predictintracount=predictintracount;
        else
            predictintracount=predictintracount+1;
        end
    end

    dataintra=intracount+predictintracount;
    sizeframe(i)=dataintra*8;

    else

    filename=sprintf('%s%d.mat','InterDWT',i);
    data=load(filename);
    data=data.scanInter;
    [row col]=size(data);
    for k=1:col
        if data(1,k)==0
            intercount=intercount;
        else
            intercount=intercount+1;
        end
    end

    filename=sprintf('%s%d.mat','motiondwt',i);
    data=load(filename);
    data=data.scanmotion;
    [row col]=size(data);
        for k=1:col
            if data(1,k)==0
            motioncount=motioncount;
            else
            motioncount=motioncount+1;
            end
        end

    datainter=intercount+motioncount;
    sizeframe(i)=datainter*8;

    end
```

```
end
save('sizeframe','sizeframe');

total_data=0;
for i=1:12
    total_data=total_data+sizeframe(i);
end

save ('total_size_dwt','total_data');
%END of BIT DATA

%DECODER
%WARNING : Ganti filter, GANTI SIZE! GANTI jml koef INVZIGZAG!
aviobj=avifile('decodecoif3filename1-
12.avi','compression','none','fps',30);

sizes=[31 31;31 31;46 46;76 76;136 136;256 256]; %ganti filter, ganti!

 %load prediction Intra frame 1 sbg referensi awal
    predictIntra=load ('predictIntraDWT1');
    predictIntra=predictIntra.scanpredictIntra;
    predictIntra=invzigzag(predictIntra,1,83008);
    dequantpredictIntra=predictIntra*40;
    invdwtpredictIntra=waverec2(dequantpredictIntra,sizes,'coif3');
    predictIntra=invdwtpredictIntra;
    %load data residu setiap frame I yg di-code
    dataIntra=load ('IntraDWT1');
    dataIntra=dataIntra.scanIntra;
    dataIntra=invzigzag(dataIntra,1,83008); %reordering jadi matrix
    dequantIntra=dataIntra*40; %rescaling dengan QP=36
    invdwtIntra=waverec2(dequantIntra,sizes,'coif3'); %inv dwt
    recframe=invdwtIntra+predictIntra; %rekonstruksi dengan prediction
Intra
    figure,imshow(uint8(recframe));
    toVid=getframe;
    aviobj=addframe(aviobj,toVid);

%load motion vector 2 utk jd referensi awal frame P
     motionVector=load('motiondwt2');
    motionVector=motionVector.scanmotion;
    motionVector=invzigzag(motionVector,2,4096);
    imgComp=motionComp(recframe,motionVector,4);

%decode frame I
for(i=2:12)
    %decode frame I
    if i==1 || i==5 || i==9
    %load prediction Intra frame 1 sbg referensi awal
    filename=sprintf('%s%d.mat','predictIntraDWT',i);
    predictIntra=load (filename);
    predictIntra=predictIntra.scanpredictIntra;
    predictIntra=invzigzag(predictIntra,1,83008);
    dequantpredictIntra=predictIntra*40;
    invdwtpredictIntra=waverec2(dequantpredictIntra,sizes,'coif3');
    predictIntra=invdwtpredictIntra;
    %load data residu setiap frame I yg di-code
    filename=sprintf('%s%d.mat','IntraDWT',i);
    dataIntra=load (filename);
    dataIntra=dataIntra.scanIntra;
    dataIntra=invzigzag(dataIntra,1,83008); %reordering jadi matrix
    dequantIntra=dataIntra*40; %rescaling dengan QP=36
    invdwtIntra=waverec2(dequantIntra,sizes,'coif3'); %inv dwt
    recframe=invdwtIntra+predictIntra; %rekonstruksi dengan prediction
Intra
    figure,imshow(uint8(recframe));
```

```
    toVid=getframe;
    aviobj=addframe(aviobj,toVid);

    %decode frame P
    else
    filename=sprintf('%s%d.mat','InterDWT',i);
    dataInter=load (filename);
    dataInter=dataInter.scanInter;
    dataInter=invzigzag(dataInter,1,83008);
    dequantInter=dataInter*40;
    invdwtInter=waverec2(dequantInter,sizes,'coif3');
    recframe=invdwtInter+imgComp;

    figure,imshow(uint8(recframe));
    toVid=getframe;
    aviobj=addframe(aviobj,toVid);
    end

     if i<12 && i~=4 && i~=8
    filename=sprintf('%s%d.mat','motiondwt',i+1);
    motionVector=load(filename);
    motionVector=motionVector.scanmotion;
    motionVector=invzigzag(motionVector,2,4096);

    mbSize=4;
    imgComp=motionComp(recframe,motionVector,mbSize);
     end


end

aviobj=close(aviobj);
 %END of DECODER

%HITUNG COMP RATIO
%size original video
info=aviinfo('filename1-12.avi');
size=info.FileSize;
sizebit=size*8;

%rasio kompresi DWT
size_dwt=load('total_size_dwt');
size_dwt=size_dwt.total_data;
compratio_dwt=sizebit/size_dwt

%rasio kompresi tiap frame
sizeframe=load('sizeframe');
sizeframe=sizeframe.sizeframe;
compframe=zeros(1,12);
for i=1:12
    compframe(i)=(sizebit/12)/sizeframe(i);
end
save('compframedwt','compframe');
figure,plot(compframe);
saveas(gcf,'CRdwtcoif3 filename1-12.fig');
%END of COMP RATIO

%HITUNG PSNR
vidOriginal=mmreader('filename1-12.avi');
nFrame=vidOriginal.NumberOfFrames;

vidDecoded=mmreader('decodecoif3filename1-12.avi');
nFrames=vidDecoded.NumberOfFrames;

psnrDWT=zeros([1 nFrame]);
```

```
for i=1:nFrame
    frameOri=read(vidOriginal,i);
    frameOri=rgb2ycbcr(frameOri);
    frameOri=double(frameOri(:,:,1));

    frameDecode=read(vidDecoded,i);
    frameDecode=rgb2ycbcr(frameDecode);
    frameDecode=double(frameDecode(:,:,1));

    psnrDWT(i)=imgPSNR(frameOri,frameDecode,255);
    figure,subplot(1,2,1),imshow(uint8(frameOri)),title(['Original frame
ke-' ,num2str(i)]);
    subplot(1,2,2),imshow(uint8(frameDecode)),title(['Decoded PSNR=',...
        num2str(psnrDWT(i)),'dB']);
    name=sprintf('%s%d.jpg','PSNRcoif3 filename',i);
    saveas(gcf,name);
end
save('psnrDWT','psnrDWT');
figure,plot(psnrDWT);
saveas(gcf,'PSNRcoif3 filename.jpg');
%END of PSNR

toc;
```

- Fungsi algoritma pencocokan blok *Three Step Search* (Aroh Barjatya, 2004)

```
function [motionVect, TSScomputations] = motionEstTSS(imgP, imgI, mbSize,
p)

[row col] = size(imgI);

vectors = zeros(2,row*col/mbSize^2);
costs = ones(3, 3) * 65537;


L = floor(log10(p+1)/log10(2));
stepMax = 2^(L-1);


mbCount = 1;
for i = 1 : mbSize : row-mbSize+1
    for j = 1 : mbSize : col-mbSize+1


        x = j;
        y = i;


        costs(2,2) = costFuncMAD(imgP(i:i+mbSize-1,j:j+mbSize-1), ...
                                 imgI(i:i+mbSize-1,j:j+mbSize-
1),mbSize);

        stepSize = stepMax;

        while(stepSize >= 1)

            % m is row(vertical) index
            % n is col(horizontal) index
            % this means we are scanning in raster order
            for m = -stepSize : stepSize : stepSize
                for n = -stepSize : stepSize : stepSize
                    refBlkVer = y + m;   % row/Vert co-ordinate for ref
block
```

```
                    refBlkHor = x + n;    % col/Horizontal co-ordinate
                    if ( refBlkVer < 1 || refBlkVer+mbSize-1 > row ...
                        || refBlkHor < 1 || refBlkHor+mbSize-1 > col)
                        continue;
                    end


                    costRow = m/stepSize + 2;
                    costCol = n/stepSize + 2;
                    if (costRow == 2 && costCol == 2)
                        continue
                    end
                    costs(costRow, costCol ) =
costFuncMAD(imgP(i:i+mbSize-1,j:j+mbSize-1), ...
                        imgI(refBlkVer:refBlkVer+mbSize-1,
refBlkHor:refBlkHor+mbSize-1), mbSize);

  %                  computations = computations + 1;
                end
            end


            [dx, dy, min] = minCost(costs);      % finds which macroblock
in imgI gave us min Cost


            % shift the root for search window to new minima point

            x = x + (dx-2)*stepSize;
            y = y + (dy-2)*stepSize;


            stepSize = stepSize / 2;
            costs(2,2) = costs(dy,dx);

        end
        vectors(1,mbCount) = y - i;     % row co-ordinate for the vector
        vectors(2,mbCount) = x - j;     % col co-ordinate for the vector
        mbCount = mbCount + 1;
        costs = ones(3,3) * 65537;
    end
end

motionVect = vectors;
```

- Fungsi *image compensation* (Aroh Barjatya, 2004)

```
function imgComp = motionComp(imgI, motionVect, mbSize)

[row col] = size(imgI);


imageComp=zeros(size(imgI));
mbCount = 1;
for i = 1:mbSize:row-mbSize+1
    for j = 1:mbSize:col-mbSize+1

        % dy is row(vertical) index
        % dx is col(horizontal) index
        % this means we are scanning in order

        dy = motionVect(1,mbCount);
        dx = motionVect(2,mbCount);
```

```
        refBlkVer = i + dy;
        refBlkHor = j + dx;
        imageComp(i:i+mbSize-1,j:j+mbSize-1) =
imgI(refBlkVer:refBlkVer+mbSize-1, refBlkHor:refBlkHor+mbSize-1);

        mbCount = mbCount + 1;
    end
end

imgComp = imageComp;
```

- Fungsi matriks untuk *forward* dan *inverse* DCT

```
function y=dct264(x)

a=[0.5 0.5 0.5 0.5;0.65 0.27 -0.27 -0.65;...
    0.5 -0.5 -0.5 0.5;0.27 -0.65 0.65 -0.27];
y=a*x*a';


function x=idct264(y)

a=[0.5 0.5 0.5 0.5;0.65 0.27 -0.27 -0.65;...
    0.5 -0.5 -0.5 0.5;0.27 -0.65 0.65 -0.27];
x=a'*y*a;
```

- Fungsi prediksi intra mode DC

```
function predictBlock=modeDC(currentBlock)

predictBlock=zeros(size(currentBlock));
%predictBlock=zeros(size(currentBlock));
for i=1:5
    for j=1:5
        predictBlock(i,j)=currentBlock(i,j);
    end
end

mean=round((predictBlock(1,2)+predictBlock(1,3)+predictBlock(1,4)+...
    predictBlock(1,5)+predictBlock(2,1)+predictBlock(3,1)+...
    predictBlock(4,1)+predictBlock(5,1)+4)/8);

for i=2:5
    for j=2:5
        predictBlock(i,j)=mean;
    end
end
```

- Fungsi zigzag *scanning* dan *inverse scanning* (Oluwadamilola M. O., 2010)

```
function out=zigzag(in)

[num_rows num_cols]=size(in);

% Initialise the output vector
out=zeros(1,num_rows*num_cols);

cur_row=1;  cur_col=1;  cur_index=1;
```

```
% First element
%out(1)=in(1,1);

while cur_row<=num_rows & cur_col<=num_cols
    if cur_row==1 & mod(cur_row+cur_col,2)==0 & cur_col~=num_cols
        out(cur_index)=in(cur_row,cur_col);
        cur_col=cur_col+1;                          %move right at the
top
        cur_index=cur_index+1;

    elseif cur_row==num_rows & mod(cur_row+cur_col,2)~=0 &
cur_col~=num_cols
        out(cur_index)=in(cur_row,cur_col);
        cur_col=cur_col+1;                          %move right at the
bottom
        cur_index=cur_index+1;

    elseif cur_col==1 & mod(cur_row+cur_col,2)~=0 & cur_row~=num_rows
        out(cur_index)=in(cur_row,cur_col);
        cur_row=cur_row+1;                          %move down at the
left
        cur_index=cur_index+1;

    elseif cur_col==num_cols & mod(cur_row+cur_col,2)==0 &
cur_row~=num_rows
        out(cur_index)=in(cur_row,cur_col);
        cur_row=cur_row+1;                          %move down at the
right
        cur_index=cur_index+1;

    elseif cur_col~=1 & cur_row~=num_rows & mod(cur_row+cur_col,2)~=0
        out(cur_index)=in(cur_row,cur_col);
        cur_row=cur_row+1;      cur_col=cur_col-1;  %move diagonally left
down
        cur_index=cur_index+1;

    elseif cur_row~=1 & cur_col~=num_cols & mod(cur_row+cur_col,2)==0
        out(cur_index)=in(cur_row,cur_col);
        cur_row=cur_row-1;      cur_col=cur_col+1;  %move diagonally
right up
        cur_index=cur_index+1;

    elseif cur_row==num_rows & cur_col==num_cols    %obtain the bottom
right element
        out(end)=in(end);                           %end of the operation
        break                                       %terminate the
operation
    end
end


function out=invzigzag(in,num_rows,num_cols)

tot_elem=length(in);

if nargin>3
    error('Too many input arguments');
elseif nargin<3
    error('Too few input arguments');
end

% Check if matrix dimensions correspond
if tot_elem~=num_rows*num_cols
    error('Matrix dimensions do not coincide');
```

```
end


% Initialise the output matrix
out=zeros(num_rows,num_cols);

cur_row=1;  cur_col=1;  cur_index=1;

% First element
%out(1,1)=in(1);

while cur_index<=tot_elem
    if cur_row==1 & mod(cur_row+cur_col,2)==0 & cur_col~=num_cols
        out(cur_row,cur_col)=in(cur_index);
        cur_col=cur_col+1;                          %move right at the
top
        cur_index=cur_index+1;

    elseif cur_row==num_rows & mod(cur_row+cur_col,2)~=0 &
cur_col~=num_cols
        out(cur_row,cur_col)=in(cur_index);
        cur_col=cur_col+1;                          %move right at the
bottom
        cur_index=cur_index+1;

    elseif cur_col==1 & mod(cur_row+cur_col,2)~=0 & cur_row~=num_rows
        out(cur_row,cur_col)=in(cur_index);
        cur_row=cur_row+1;                          %move down at the
left
        cur_index=cur_index+1;

    elseif cur_col==num_cols & mod(cur_row+cur_col,2)==0 &
cur_row~=num_rows
        out(cur_row,cur_col)=in(cur_index);
        cur_row=cur_row+1;                          %move down at the
right
        cur_index=cur_index+1;

    elseif cur_col~=1 & cur_row~=num_rows & mod(cur_row+cur_col,2)~=0
        out(cur_row,cur_col)=in(cur_index);
        cur_row=cur_row+1;      cur_col=cur_col-1;  %move diagonally left
down
        cur_index=cur_index+1;

    elseif cur_row~=1 & cur_col~=num_cols & mod(cur_row+cur_col,2)==0
        out(cur_row,cur_col)=in(cur_index);
        cur_row=cur_row-1;      cur_col=cur_col+1;  %move diagonally
right up
        cur_index=cur_index+1;

    elseif cur_index==tot_elem                      %input the bottom
right element
        out(end)=in(end);                           %end of the operation
        break                                       %terminate the
operation
    end
end
```

- Fungsi untuk perhitungan PSNR

```
function psnr = imgPSNR(imgP, imgComp, n)

[row col] = size(imgP);
```

```
err = 0;

for i = 1:row
    for j = 1:col
        err = err + (imgP(i,j) - imgComp(i,j))^2;
    end
end
mse = err / (row*col);

psnr = 10*log10(n*n/mse);
```