BAB 2

LANDASAN TEORI

2.1 *System Development Life Cycle* (SDLC)

System Development Life Cycle (SDLC) dapat dianggap sebagai kerangka kerja formal tertua metodologi untuk membangun sistem informasi. Ide utama dari SDLC adalah "untuk mengejar pengembangan sistem informasi dalam cara yang terstruktur dan metodis, yang mengharuskan tahap *life cycle* dari mulai ide awal sampai pada pengiriman tahap final sistem, untuk dilaksanakan secara beraturan". Salah satu tipe SDLC yang paling awal dan paling banyak digunakan adalah metode *Waterfall*.

Waterfall method sering dianggap sebagai pendekatan klasik dengan siklus hidup pengembangan sistem. Pembangunan dengan metode *Waterfall* memiliki tujuan yang berbeda untuk setiap fase pembangunan. Setelah fase pembangunan selesai, hasil pengembangan ke tahap berikutnya dan tidak ada jalan kembali.

Keuntungan dari pembangunan air terjun adalah bahwa hal itu memungkinkan untuk departmentalization dan kontrol manajerial. Sebuah jadwal bisa diatur dengan tenggat waktu untuk setiap tahap pengembangan dan produk dapat dilanjutkan melalui proses pengembangan seperti mobil di carwash, dan secara teoritis, akan dikirimkan tepat waktu.

Kerugian dari pembangunan air terjun adalah bahwa hal itu tidak memungkinkan untuk banyak refleksi atau revisi. Setelah aplikasi adalah dalam tahap pengujian, sangat sulit untuk kembali dan mengubah sesuatu yang tidak

dipikirkan baik-dalam tahap konsep. Alternatif untuk model air terjun termasuk pengembangan aplikasi bersama (JAD), pengembangan aplikasi cepat (RAD), selaras dan menstabilkan, membangun dan memperbaiki, dan model spiral. Tahapan-tahapan pada metode *Waterfall*:

1. ***Requirement Analysis***

   Seluruh kebutuhan software harus bisa didapatkan dalam fase ini, termasuk didalamnya kegunaan software yang diharapkan pengguna dan batasan software. Informasi ini biasanya dapat diperoleh melalui wawancara, survey atau diskusi. Informasi tersebut dianalisis untuk mendapatkan dokumentasi kebutuhan pengguna untuk digunakan pada tahap selanjutnya.

2. ***System Design***

   Tahap ini dilakukan sebelum melakukan coding. Tahap ini bertujuan untuk memberikan gambaran apa yang seharusnya dikerjakan dan bagaimana tampilannya. Tahap ini membantu dalam menspesifikasikan kebutuhan hardware dan sistem serta mendefinisikan arsitektur sistem secara keseluruhan.

3. ***Implementation***

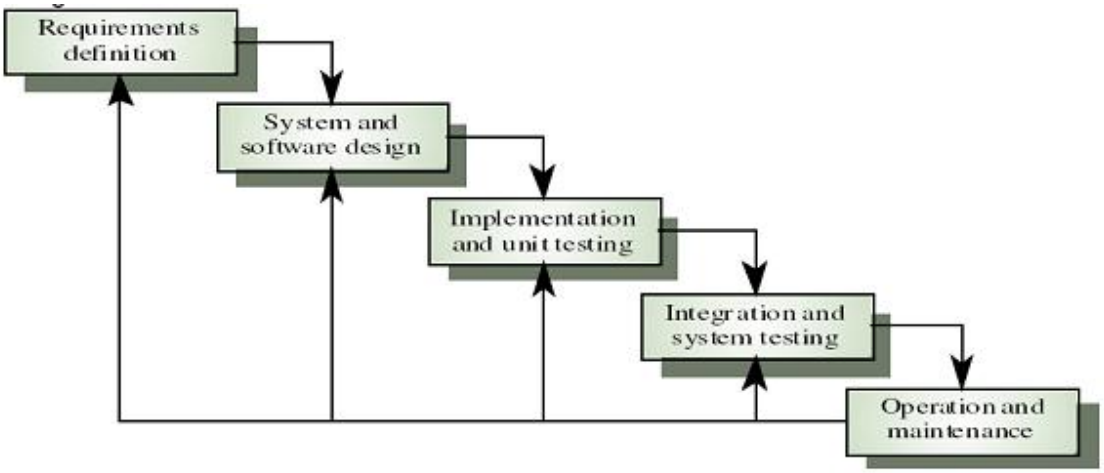
   Dalam tahap ini dilakukan pemrograman. Pembuatan software dipecah menjadi modul-modul kecil yang nantinya akan digabungkan dalam tahap berikutnya. Selain itu dalam tahap ini juga dilakukan pemeriksaaan terhadap modul yang dibuat, apakah sudah memenuhi fungsi yang diinginkan atau belum.

4. ***Integration & Testing***

Di tahap ini dilakukan penggabungan modul-modul yang sudah dibuat dan dilakukan pengujian ini dilakukan untuk mengetahui apakah software yang dibuat telah sesuai dengan desainnya dan masih terdapat kesalahan atau tidak.

5. ***Operation & Maintenance***

Ini merupakan tahap terakhir dalam model waterfall. Software yang sudah jadi dijalankan serta dilakukan pemeliharaan. Pemeliharaan termasuk dalam memperbaiki kesalahan yang tidak ditemukan pada langkah sebelumnya. Perbaikan implementasi unit sistem dan peningkatan jasa sistem sebagai kebutuhan baru.



Gambar 2.1 *Waterfall Method*

2.2 Steganografi

Kata steganografi (*Steganography*) berasal dari kata Yunani. *Steganos* yang artinya ‘tersembunyi/terselubung’, dan *graphien*, ‘menulis’ sehingga kurang lebih artinya “menulis (tulisan) terselubung”. (Budi, 2002)

Steganografi adalah suatu teknik untuk menyembunyikan informasi yang bersifat pribadi dengan sesuatu yang hasilnya akan tampak seperti informasi normal lainnya. Media yang digunakan umumnya merupakan suatu media yang berbeda dengan media pembawa informasi rahasia, dimana disinilah fungsi dari teknik steganografi yaitu sebagai teknik penyamaran menggunakan media lain yang berbeda sehingga informasi rahasia dalam media awal tidak terlihat secara jelas. Steganografi juga berbeda dengan kriptografi yaitu terletak pada hasil dari prosesnya. Hasil dari kriptografi biasanya berupa data yang berbeda dari bentuk aslinya dan biasanya datanya seolah-olah berantakan namun dapat dikembalikan ke data semula. Sedangkan hasil dari keluaran steganografi memiliki bentuk yang sama dengan data aslinya, tentu saja persepsi ini oleh indra manusia, tetapi tidak oleh komputer atau pengolah data digital lainnya. Namun secara umum steganografi dan kriptografi mempunyai tujuan yang sama yakni mengamankan data, bagaimana supaya data tidak dapat dibaca, dimengerti atau diketahui secara langsung. Media cover merupakan data *digital* yang akan ditempeli dengan data yang akan disembunyikan atau sering disebut dengan stego medium. Berbagai media yang dapat digunakan sebagai cover dari data atau informasi yang akan disembunyikan dengan berbagai teknik steganografi.

Steganografi dapat digunakan untuk berbagai macam alasan. Untuk tujuan legitimasi dapat digunakan pengamanan seperti citra dengan watermarking dengan alasan untuk perlindungan hak cipta atau *copyright*. Steganografi juga dapat digunakan sebagai cara untuk membuat pengganti suatu nilai hash satu arah (yaitu pengguna mengambil suatu masukan panjang variabel dan membuat sebuah keluaran panjang statis dengan tipe string untuk melakukan verifikasi bahwa tidak ada perubahan yang dibuat pada variabel masukan yang asli). Selain itu juga, steganografi dapat digunakan sebagai tag-notes untuk citra online. Steganografi juga dapat digunakan untuk melakukan 3 perawatan atas kerahasiaan informasi yang berharga, untuk menjaga data tersebut dari kemungkinan sabotasi, pencuri, atau dari pihak yang tidak berwenang.

2.3 JPEG (*Joint Photographic Experts Group*)

2.3.1 Sejarah JPEG

Merupakan salah satu tipe format pada file *image*. Dikembangkan pada awal tahun 1980 oleh *Joint Photographic Experts Group*. JPEG merupakan tipe format yang paling sering digunakan di internet. Implementasi format JPEG terbaru dimulai sejak tahun 1996 dan semakin berkembang dengan inovasi format baru yang menyertai perkembangan teknologi yang memanfaatkan format JPEG lebih luas. Walaupun format JPEG merupakan metode kompresi gambar yang gratis, sebuah perusahaan bernama **Forgent** pada tahun 2002 mempatenkan format ini dan akan menarik biaya lisensi. Segera Grup JPEG mengumumkan sebuah format JPEG 2000 sebagai sebuah

format pengganti. Namun dua hal di atas terlambat, karena JPEG sudah digunakan secara luas dan hak paten belum ditetapkan oleh pengadilan.

Standar kompresi file gambar yang dibuat oleh *kelompok Joint Photographic Experts Group* ini menghasilkan kompresi yang sangat besar tetapi dengan akibat berupa adanya distorsi pada gambar yang hampir selalu tidak terlihat. JPEG adalah sebuah format gambar, sangat berguna untuk membuat gambar jenis fotografi berkualitas tinggi dalam ukuran file yang sangat kecil. Format file grafis ini telah diterima oleh *Telecommunication Standardization Sector* atau ITU-T dan Organisasi Internasional untuk Standardisasi atau ISO. JPEG kebanyakan digunakan untuk melakukan kompresi gambar diam menggunakan analisis *Discrete Cosine Transform* (DCT).

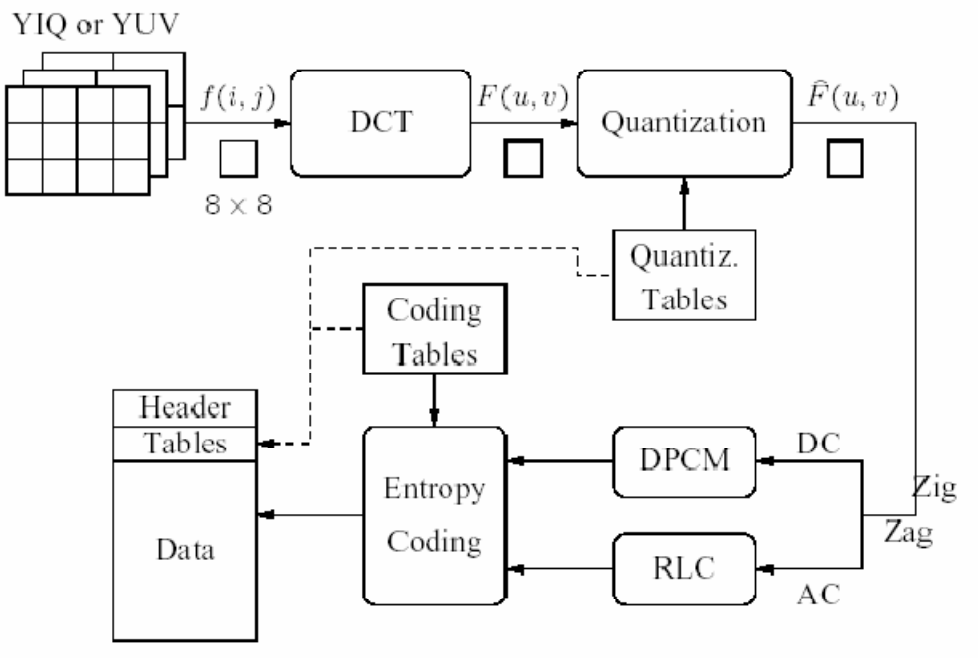
Meskipun kompresi gambar JPEG sangatlah efisien dan selalu menyimpan gambar dalam kategori warna true color (24 bit), format ini bersifat lossy, yang berarti bahwa kualitas gambar dikorbankan bila tingkat kompresi yang dipilih semakin tinggi.

2.3.2 Kompersi JPEG

Dimulai sejak beberapa tahun lalu, JPEG (*Joint Photographic Experts Group*) membuat teknik kompresi international pertama untuk format *file* citra. Pada tahun 1992 teknik kompresi ini mulai diterima secara formal sebagai standar internasional. (Leung, 2004). Standar ini ditetapkan oleh JPEG agar dapat memenuhi kebutuhan berbagai aplikasi yang bekerja dengan

file image. Kompresi yang diajukan oleh JPEG ini dapat bekerja dengan citra berwarna maupun *greyscale*.

Berikut akan dijelaskan secara lebih lanjut untuk tahapan pada kompresi JPEG:



Gambar 2.2 Tahapan dalam kompresi JPEG (Leung, 2004)

2.3.2.1 Konversi dari RGB menjadi YcbCr

RGB merupakan singkatan dari Red-Green-Blue. Tiga warna yang di jadikan patokan warna secara universal. Dengan basis RGB, kita bisa mengubah warna ke dalam kode-kode angka sehingga warna tersebut akan tampil universal. Dengan standar RGB, seorang dapat mengatakan warna dengan komposisi angka yang jelas. Format warna RGB adalah format yang digunakan dalam *file* bitmap,

dimana setiap *pixel* terdiri atas terdiri atas komposisi tiga variabel yang berisi nilai masing-masing warna.

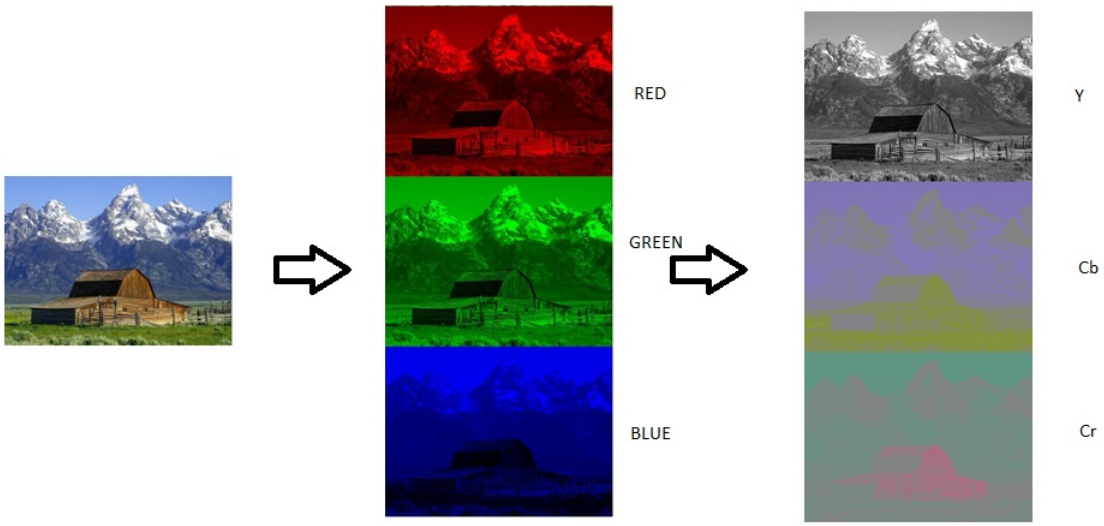
Konversi warna RGB ke dalam warna YCbCr, warna luminance atau dikenal dengan istilah grayscale, yaitu gambar dengan derajat keabuan yang mempunyai intensitas warna 0 sampai 255, dimana 0 adalah untuk merepresentasikan warna hitam dan 255 adalah warna untuk merepresentasikan warna putih. Karena mata manusia lebih sensitif pada warna luminance (Y) dari pada warna chrominance (Cb,Cr), sehingga informasi warna chrominance tidak diikut sertakan pada proses kompresi dan hanya warna Y yang diproses sebagai masukan gambar untuk proses selanjutnya. (Gunawan, 2003)

Warna YCbCr diperoleh dengan mentrransformasikan RGB dengan rumus :

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.144 \\ -0.159 & -0.332 & 0.050 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} = \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Rumus Perkalian untuk Mengubah RGB menjadi YcbCr (Leung, 2004)

Berikut adalah contoh dari gambar yang dikonversi menjadi CbCr :



Gambar 2.3 Contoh koversi RBG ke YCbCr

2.3.2.2 *Discrete Cosine Transform* (DCT)

DCT merupakan fungsi yang digunakan untuk merubah nilai YcbCr pada setiap *pixel* menjadi koefisien DCT. Dan setelah melalui beberapa kali penelitian ditemukan bahwa dengan menggunakan 8 x 8 DCT telah menghasilkan gambar dengan kualitas paling baik, namun teknik ini tetap memiliki kekurangan yaitu efek isolasi antara blok – blok *pixel* yang saling berdekatan. Inilah mengapa gambar JPEG yang dikompres dengan rasio yang tinggi akan terlihat kotak-kotak. (Leung, 2004)

Hal yang pertama kali dilakukan pada tahap DCT ini adalah membagi keseluruhan gambar menjadi 8 x 8 *pixel*. Kemudian setiap

blok-blok *pixel* tersebut diproses satu persatu menjadi 64 keofisien DCT melalui rumus :

$$F(u,v)=\frac{C(u)C(v)}{4}\sum_{i=0}^{7}\sum_{j=0}^{7}\cos\frac{(2i+1)u\pi}{16}\cos\frac{(2j+1)u\pi}{16}\ f(i,j)\ C(\xi)=\begin{cases}\frac{\sqrt{2}}{2} & jika\ \xi=0\\ 1 & lainnya\end{cases}$$

Tujuan dari tahap ini adalah karena pada gambar yang belum terkompresi nilai koefisien DCT rata-rata berukuran amat kecil dan banyak yang dapat dihilangkan dengan tetap mempertahankan keakuratan gambar.

$$\begin{bmatrix}139&144&149&153&155&155&155&155\\144&151&153&156&159&156&156&156\\150&155&160&163&158&156&156&156\\159&161&162&160&160&159&159&159\\159&160&161&162&162&155&155&155\\161&161&161&161&160&157&157&157\\162&162&161&163&162&157&157&157\\162&162&161&161&163&158&158&158\end{bmatrix}\rightarrow\begin{bmatrix}1260&-1&-12&-5&2&-2&-3&1\\-23&-17&-6&-3&-3&0&0&1\\-11&-9&-2&2&0&-1&-1&0\\-7&-2&0&1&1&0&0&0\\-1&-1&1&2&0&-1&1&1\\2&0&2&0&-1&1&1&-1\\-1&0&0&-1&0&2&1&-1\\-3&2&-4&-2&2&1&-1&0\end{bmatrix}$$

Gambar 2.4 Perubahan nilai warna menjadi koefisien DCT

Dibanding nilai 63 koefisien DCT lainnya, koefisien pertama dari tiap blok pasti memiliki nilai yang paling besar karena merupakan nilai rata-rata dari keseluruhan blok, koefisien pertama disebut koefisien DC dan 63 koefisien lainnya disebut koefisien AC. Untuk mengembalikan kembali koefisien DCT yang didapat kedalam 64 nilai *pixel* sebelumnya harus dilakukan tahap Invers

DCT, tetapi hasil yang didapat akan sedikit mengalami perubahan sehingga tahap ini dinamakan tahap *lossy*.

Rumus Invers DCT adalah sebagai berikut :

$$\tilde{f}(i,j)=\sum_{u=0}^{7}\sum_{v=0}^{7}\frac{C(u)C(v)}{4}\cos\frac{(2i+1)u\pi}{16}\cos\frac{(2j+1)v\pi}{16}F(u,v), \qquad C(\xi)\begin{cases}\frac{\sqrt{2}}{2} & jika\ \zeta=0\\ 1 & lainnya\end{cases}$$

2.3.2.3 Kuantisasi

Proses kuantisasi merupakan proses untuk mengurangi jumlah bit yang diperlukan untuk menyimpan suatu data gambar. Karena mata manusia lebih peka terhadap frekuensi rendah dari pada frekuensi tinggi dan karena frekuensi tinggi tidak merubah data gambar secara signifikan, maka pada proses kuantisasi frekuensi tinggi ini dipotong dengan cara, matriks koefesien hasil DCT dibagi dengan matriks *quantum*. Matriks quantum ini ditentukan oleh faktor kualitas yang dipilih antara 1 sampai 100 yang nantinya dipakai untuk menentukan kualitas dari suatu gambar JPEG.

Tahap kuantisasi juga merupakan tahap lossy dalam kompresi JPEG karena kuantisasi melakukan pembagian antara setiap koefisien DCT dengan koefisien dari matriks *quantum* yang ditentukan dan melakukan pembulatan setelahnya.

Berikut adalah tabel kuantisasi untuk koefisien *luminance* dan *chrominance* yang sudah ditetapkan sebagai standar oleh JPEG

dengan rasio kompresi paling baik dan penurunan kualitas gambar paling rendah :

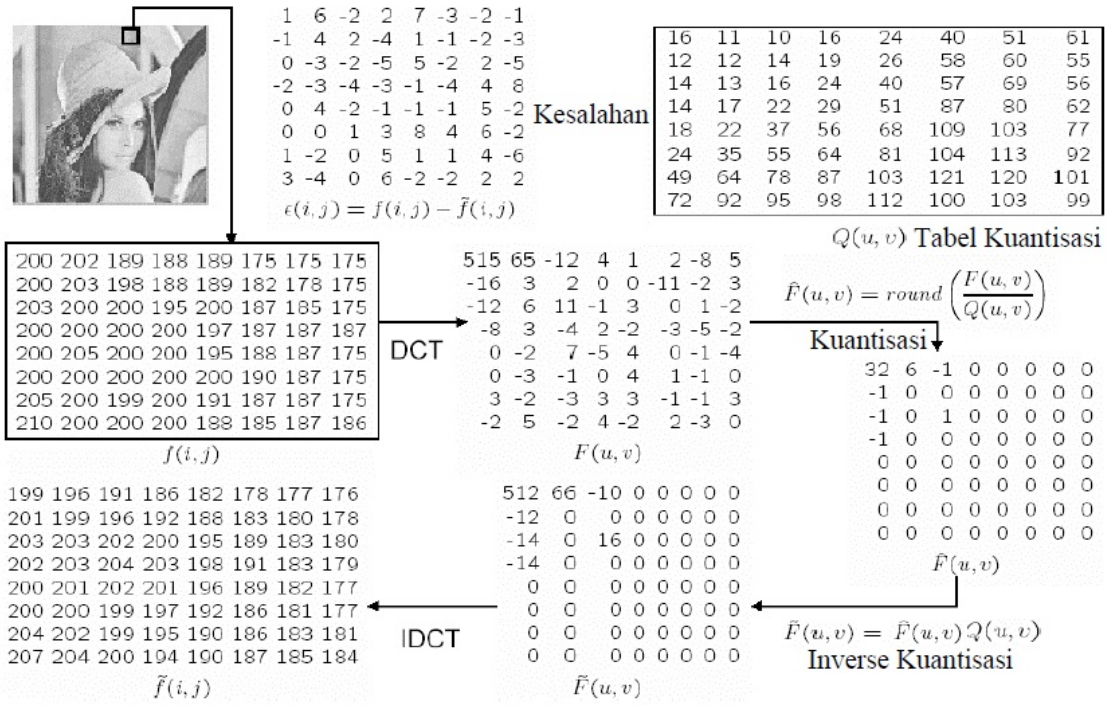
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

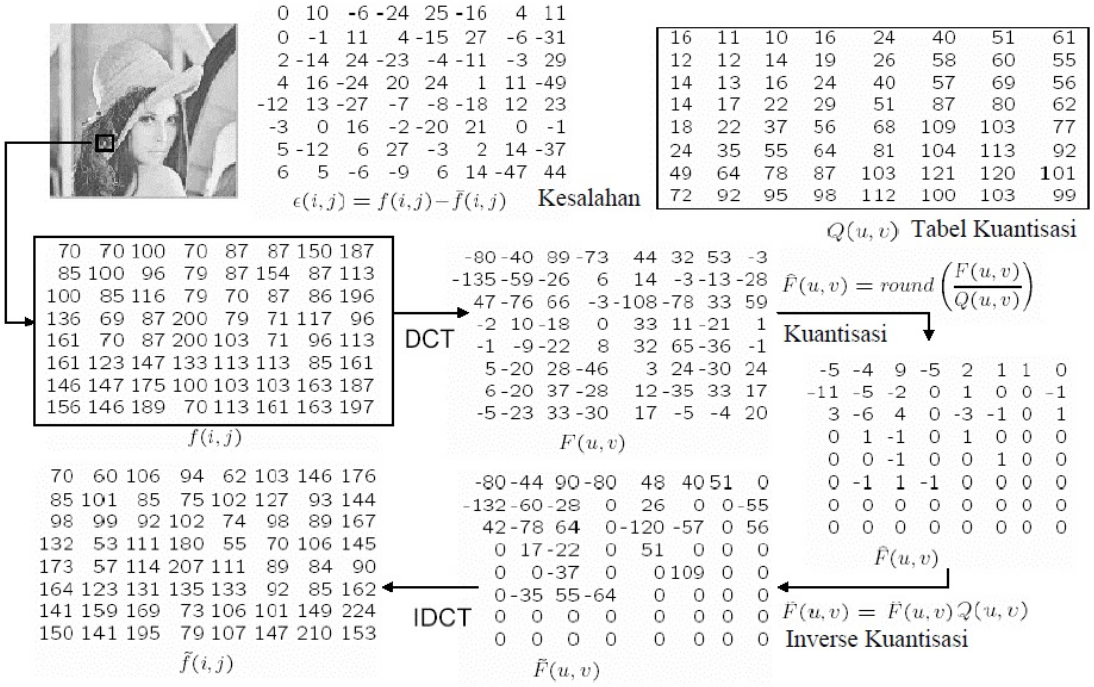
Gambar 2.5 Tabel Kuantisasi *Luminence* dan Table Kuantisasi *Chrominence*

(Leung, 2004)

Keragaman warna pada suatu blok ternyata juga sangat berpengaruh pada penurunan kualitas yang disebabkan oleh dua tahap *lossy* ini (DCT dan Kuantisasi). Berikut contoh dari seberapa besar pengaruh keragaman warna pada penurunan kualitas gambar :



Gambar 2.6 Perubahan nilai warna dengan warna cenderung seragam (Leung, 2004)

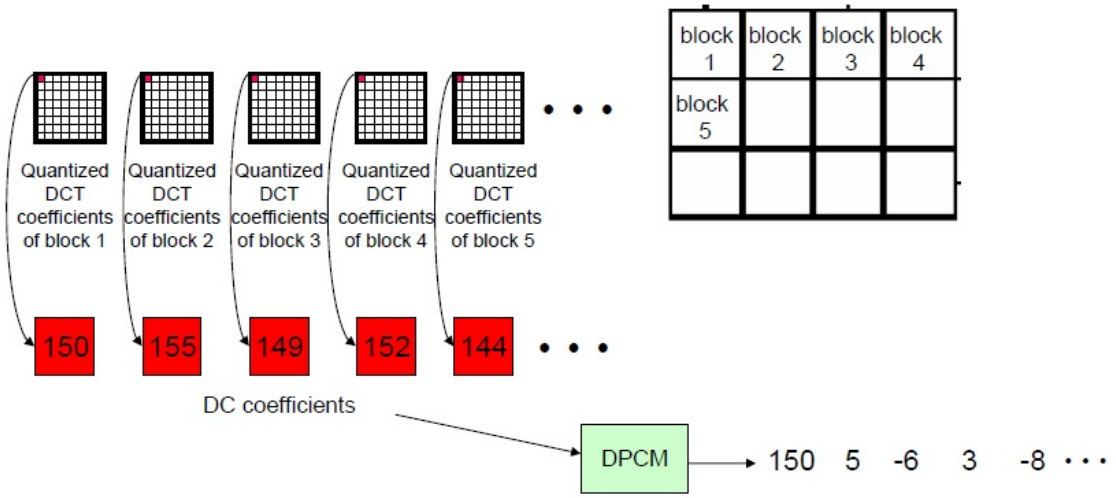


Gambar 2.7 Perubahan nilai warna dengan warna variatif (Leung, 2004)

Dapat kita lihat kesalahan yang terjadi pada blok dengan tekstur warna yang beragam lebih besar dibanding dengan blok yang mempunyai tekstur warna relatif sama.

2.3.2.4 *Differential Pulse Code Modulation* (DCPM)

Pada tahap ini, koefisien DC dari tiap blok disatukan untuk memasuki tahap *Entropy Coding*, teknik DPCM digunakan karena nilai-nilai koefisien DC antar blok tidak berbeda jauh. (Leung, 2004)



Gambar 2.8 *Differential Pulse Code modulation*

2.3.2.5 *Entropy coding* pada koefisien DC

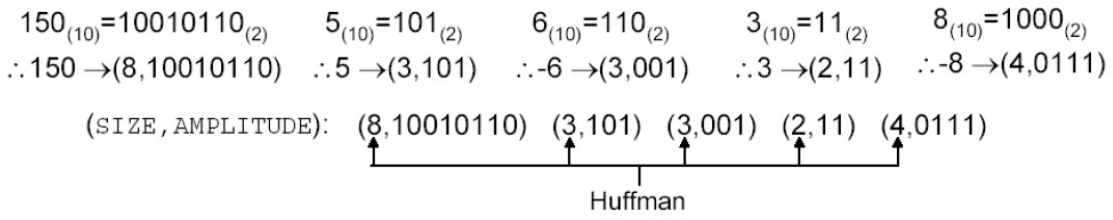
Koefisien DC yang sudah melalui tahap DPCM kemudian dikompresi menggunakan metode *Huffman*, tetapi sebelumnya deretan angka tersebut akan dirubah bentuknya menjadi pasangan-pasangan (*size, amplitude*) dimana *size* menyatakan jumlah bit yang

diperlukan untuk merepresentasikan jumlah angka DPCM dan *amplitude* menyatakan angka tersebut dalam bit (Leung, 2004). Berikut tabel yang menyatakan hubungan antara *size* dan *amplitude*:

Tabel 2.1 Hubungan *size*, *amplitude*, dan *number* (Leung, 2004)

SIZE	AMPLITUDE	NUMBER
1	0,1	-1,1
2	00,01,10,11	-3,-2,2,3
3	000,...,011,100,...,111	-7,...,-4,4,...,7
4	0000,...,0111,1000,...,111	-15,...,-8,8,...,15
...	...	...
10	0000000000,...,0111111111,1000000000,...,1111111111	-1023,...,-512,512,...,1023

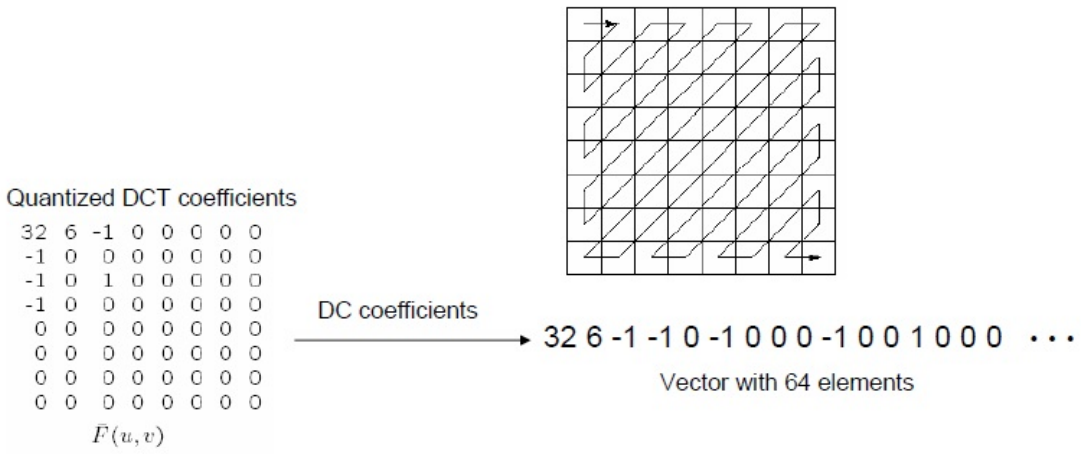
Dalam *Entropy coding* yang mengalami kompresi *huffman* hanya *size*-nya saja, karena perubahan *size* tidak terlalu jauh sedangkan *amplitude*-nya bervariasi.



Gambar 2.9 Proses *Entropy Encoding* pada koefisien DC (Leung, 2004)

2.3.2.6 *Zig-zag Scanning*

Zig-zag scanning yaitu proses yang merubah matriks 8 x 8 hasil proses kuantisasi kedalam vektor 1 x 2^8 , dengan pembacaan secara zig-zag scanning. Pada proses zig-zag scanning ini keofisien DCT terkuantisasi yang bernilai nol cenderung terbaca secara berurutan.



Gambar 2.10 Proses *Zig-Zag Scan*

2.3.2.7 *Run Length Code*

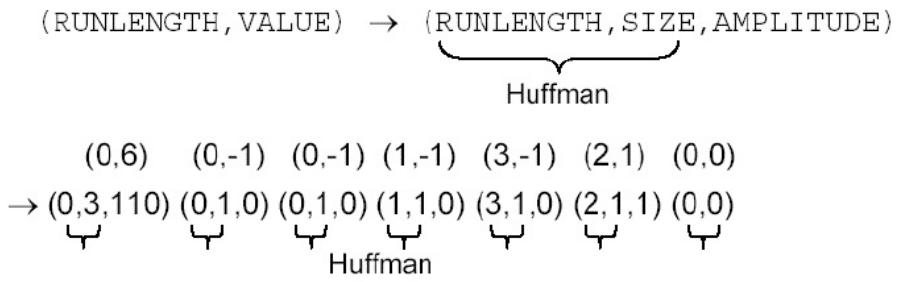
RLC (Run-Length Code) yaitu proses serangkaian simbol yang berurutan dikodekan menjadi suatu kode yang terdiri dari simbol tersebut dan jumlah pengulangannya. RLC efektif karena hasil keluaran matriks setelah proses kuantisasi pada frekuensi tinggi cenderung nol (0) dan berurutan, Karena hampir setengahnya lebih adalah nol, maka nilai 0 inilah yang disimbolkan menjadi 0 dan jumlah pengulangannya. Untuk proses dekompresi, dilakukan proses

sebaliknya yaitu hasil pengkodean RLC discan dan diuraikan kembali, kemudian kode hasil penguraian dibaca sebagai blok. Berikut ini adalah contoh perubahan proses setelah proses zig-zag scanning ke dalam proses RLC.

Setelah diubah urutannya, niai AC kemudian diubah bentuknya menjadi pasangan-pasangan (*runlength,value*), dimana *runlength* adalah jumlah 0 yang berurutan dan *value* adalah nilai non – 0 yang terletak sesudahnya. Dalam hal ini koefisien DC tidak diperhitungkan dalam RLC.

2.3.2.8 *Entropy coding* pada koefisien AC

Koefisien AC yang sudah melalui tahap RLC juga dikompresi menggunakan kompresi *huffman*, pasangan-pasangan sebelumnya diubah lagi menjadi pasangan-pasangan (*runlength, size, value*). Dalam hal ini yang mengalami kompresi *huffman* hanya *runlength* dan *size*-nya seperti pada koefisien DC. Lebih lanjut dijelaskan dalam gambar :



Gambar 2.11 Proses *Entropy Encoding* pada koefisein AC (Leung, 2004)

Ide yang baik ketika melakukan apapun transaksi sensitif, seperti pembelian online kartu kredit, atau diskusi tentang rahasia perusahaan antara berbagai departemen dalam organisasi. Semakin kuat cipher - yaitu, sulit bagi orang yang tidak berhak untuk istirahat itu - lebih baik, secara umum. Namun, sebagai kekuatan enkripsi/dekripsi meningkat, begitu juga biaya.

Dalam beberapa tahun terakhir, kontroversi telah muncul lebih dari apa yang disebut enkripsi yang kuat. Hal ini mengacu pada cipher yang pada dasarnya bisa dipecahkan tanpa kunci dekripsi. Sementara kebanyakan perusahaan dan pelanggan mereka melihatnya sebagai suatu cara menjaga rahasia dan meminimalkan penipuan, beberapa pemerintah melihat enkripsi yang kuat sebagai kendaraan potensial dimana teroris mungkin menghindari pihak berwenang. Pemerintah ini, termasuk dari Amerika Serikat, ingin mendirikan suatu pengaturan kunci-escrow. Ini berarti setiap orang yang menggunakan sandi akan diminta untuk memberikan pemerintah dengan salinan kunci. Kunci dekripsi akan disimpan di tempat yang seharusnya aman, hanya digunakan oleh pihak berwenang, dan digunakan hanya jika didukung oleh perintah pengadilan. Lawan dari skema ini berpendapat bahwa penjahat bisa hack ke database kunci escrow dan secara ilegal memperoleh, mencuri, atau mengubah tombol. Pendukung mengklaim bahwa sementara ini kemungkinan, mengimplementasikan skema escrow

tombol akan lebih baik daripada tidak melakukan apapun untuk mencegah penjahat dari bebas menggunakan enkripsi/dekripsi.

2.4 Aplikasi *Mobile*

Aplikasi *mobile,* seperti halnya aplikasi pada sistem computer, yaitu program yang digunakan untuk melakukan suatu perintah. Hanya saja perintah ini dilakukan pada perangkat *mobile*. Mobile dapat diartikan sebagai perpindahan yang mudah dari satu tempat ke tempat yang lain, misalnya telepon mobile berarti bahwa terminal telepon yang dapat berpindah dengan mudah dari satu tempat ke tempat lain tanpa terjadi pemutusan atau terputusnya komunikasi (Romdini, 2010).

Sistem aplikasi mobile adalah aplikasi yang dapat digunakan pengguna dengan berpindah-pindah tempat dengan mudah dari suatu tempat ke tempat lain tanpa pemutusan atau terputusnya komunikasi. Contoh perangkat mobile misalnya telepon genggam, PDA, maupun tablet PC.

2.5 Java

Menurut Hombar (2010), Java adalah sebuah teknologi yang diperkenalkan oleh *Sun Microsystems* pada pertengahan tahun 1990. Menurut definisi dari Sun, *Java* adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *standalone* ataupun pada lingkungan jaringan. Kita lebih menyukai menyebut *Java* sebagai sebuah teknologi dibanding hanya sebuah

bahasa pemrograman, karena *Java* lebih lengkap dibanding sebuah bahasa pemrograman konvensional.

Java adalah bahasa pemrograman yang berorientasi objek (OOP). Perkembangan *Java* tidak hanya terfokus oada satu sistem operasi, tetapi dikembangkan untuk berbagai sistem operasi dan bersifat *open source*, karena itu kelebihan utama dari *Java* ialah dapat dijalankan di beberapa platform / sistem operasi komputer, sesuai dengan prinsip tulis sekali, jalankan di mana saja.

Beberapa kelebihan *Java* menurut *Sun Microsystem* yaitu :

a. Sederhana (*Simple*), *Java* dimodelkan sebagian dari bahasa C++ dengan memperbaiki beberapa karakteristik C++, seperti penambahan fungsionalitas, pengurangan kompleksitas. Contohnya, *Java* menghilangkan *multiple inheritance* dari C++ dengan menggunakan *interface*.
b. Berorientasi Objek (*Object Oriented*), dalam memecahkan masalah, bahasa *Java* membagi program menjasi objek-objek, kemudian memodelkan sifat dan tingkah laku masing-masing objek. Kemudian *Java* menentukan dan mengatur interaksi antar objek.
c. Terdistribusi (*Distributed*), fitur- fitur *Java* sangat mendukung teknologi internet yang saat ini berkembangsehingga dapat mendukung pemrograman terdistribusi.
d. *Multiplatform*, Bahasa *Java* dapat diterjemahkan oleh *java interpreter* pada berbagai macam sistem operasi (linux, windows, apple)

e. *Multithreaded*, *thread* adalah proses yang dapat dikerjakan oleh suatu program dalam suatu waktu. *Java* bersifat *multithreaded*, sehingga dapat mengerjakan beberapa proses dengan waktu yang hampir bersamaan.

2.6 *Android*

Android adalah sistem operasi yang digunakan di *smartphone* dan juga tablet PC. Fungsinya sama seperti sistem operasi Symbian di Nokia, iOS di Apple dan BlackBerry OS. Tetapi *Android* tidak terikat ke satu merek *Handphone* saja, beberapa vendor terkenal yang sudah memakai Android antara lain Samsung , Sony Ericsson, HTC, Nexus, Motorolla, dan lain-lain.

Android pertama kali dikembangkan oleh perusahaan bernama *Android Inc.*, dan pada tahun 2005 di akuisisi oleh raksasa Internet *Google*. *Android* dibuat dengan basis kernel Linux yang telah dimodifikasi, dan untuk setiap *release*-nya diberi kode nama berdasarkan nama hidangan makanan.

Keunggulan utama *Android* adalah gratis dan *open source*, yang membuat *smartphone Android* dijual lebih murah dibandingkan dengan Blackberry atau iPhone meski fitur (*hardware*) yang ditawarkan Android lebih baik.

2.7 Algoritma Steganografi pada *File* JPEG

2.7.1 Algoritma Jsteg

Algoritma ini dibuat oleh Derek Upham dan merupakan algortima steganografi pertama yang dipublikasikan untuk format JPEG. Algortima ini menyisipkan data kedalam koefisien DCT JPEG secara sekuensial dengan mengganti LSB dengan LSB dari data yang disisipkan. *File*

JPEG, tidak terdeteksi dengan serangan *visual* karena tidak mengubah struktur *visual*. Melainkan kefisien DCT dan cenderung berukuran lebih kecil dari BMP, oleh karena itu sangan populer di internet dan mengalami perkembangan pesat dalam hal algortima steganografi.

2.7.2 Algoritma F5

Algoritma steganografi F5 diperkenalkan oleh Pfitzmann peneliti Jerman dan Westfeld pada tahun 2001. Tujuan dari mereka penelitian ini adalah untuk mengembangkan konsep dan metode embedding praktis untuk gambar JPEG yang akan memberikan steganografi tinggi kapasitas tanpa mengorbankan keamanan. Daripada mengganti LSB dari koefisien DCT terkuantisasi dengan bit pesan, nilai absolut dari koefisien menurun satu. Para penulis berpendapat bahwa jenis *embedding* tidak dapat dideteksi menggunakan serangan statistik $\chi 2$. Algoritma F5 meng-*embed* bit pesan ke koefisien DCT yang dipilih secara acak dan menggunakan matriks *embedding* yang meminimalkan jumlah perubahan yang perlu untuk menanamkan panjang pesan tertentu.

Dalam proses *embedding*, panjang pesan dan jumlah non-nol non-DC koefisien yang digunakan untuk menentukan matriks *embedding* terbaik yang meminimalkan jumlah modifikasi dari gambar cover. *Embedding Matrix* memiliki tiga parameter (*c, n, k*), dimana *c* adalah jumlah perubahan per kelompok koefisien *n*, dan *k* adalah jumlah bit tertanam. Dalam tulisan mereka, penulis menggambarkan matriks

sederhana embedding (*1, 2k-1, k*) menggunakan "hash" fungsi yang menghasilkan *k* bit bila diterapkan pada *2k- 1* koefisien.

Proses *embedding* dimulai dengan menurunkan benih untuk PRNG (*Pseudo Random Number Generator*) dari kata sandi pengguna dan menghasilkan "*random walk*" koefisien DCT dari *cover image* tersebut. PRNG juga digunakan untuk mengenkripsi nilai *k* menggunakan *stream cipher* dan menanamkannya dalam cara yang teratur bersama-sama dengan panjang pesan di awal aliran pesan. Tubuh pesan tertanam menggunakan *embedding* matriks, menyisipkan *k* bit pesan ke satu kelompok *2k-1* koefisien dengan menurunkan nilai absolut paling banyak satu koefisien dari masing-masing kelompok satu.

Proses embedding terdiri dari langkah-langkah berikut:

a. Ambil nilai RGB dari gambar input
b. Hitung tabel kuantisasi yang sesuai dengan faktor kualitas Q dan kompres gambar saat menyimpan DCT terkuantisasi koefisien.
c. Hitung perkiraan kapasitas tanpa *embedding* matriks $C = h_{DCT} - h_{DCT} / 64 - h(0) - h(1) + 0.49h(1)$, di mana h_{DCT} adalah jumlah semua koefisien DCT, $h(0)$ adalah jumlah koefisien DCT AC bernilai nol, $h(1)$ adalah jumlah dari AC Koefisien DCT dengan nilai absolut 1, $h_{DCT}/64$ adalah jumlah dari DC koefisien. Parameter *C* dan panjang pesan yang digunakan untuk menentukan matriks *embedding* terbaik.
d. Password yang ditentukan pengguna digunakan untuk menghasilkan benih untuk PRNG juga digunakan menentukan jalur acak untuk

embedding bit-bit pesan. PRNG juga digunakan untuk menghasilkan *pseudo-random bit-stream* yang diXOR dengan pesan untuk membuatnya *bit-stream* teracak. Selama *embedding*, koefisien DC dan koefisien sama dengan nol dilewati.

e. Pesan dibagi menjadi segmen-segmen dari *k* bit yang tertanam ke dalam kelompok 2^k-1 koefisien sepanjang jalur acak. Jika hash dari kelompok yang tidak cocok dengan bit-bit pesan, nilai absolut dari salah satu koefisien dalam kelompok diturunkan satu untuk mendapatkan nilai yang cocok. Jika koefisien menjadi nol, kejadian ini disebut sebagai penyusutan, dan *k* bit pesan yang sama di*embed* ulang dalam kelompok berikutnya dari koefisien DCT.

f. Jika ukuran pesan sesuai dengan perkiraan kapasitas, maka proses *embed* berlanjut, lain daripada itu *error* yang menunjukkan panjang maksimal yang mungkin akan ditampilkan.

Algoritma F5 ini tidak memodifikasi histogram koefisien DCT, Algoritma ini menunjukkan bahwa beberapa karakteristik penting dari histogram tetap dipertahankan, seperti yang kemonotonan dan kemonotonan dari kenaikan. Algoritma F5 tidak dapat dideteksi dengan menggunakan serangan $\chi 2$ karena *embedding* tidak didasarkan pada penggantian bit maupun pertukaran nilai tetap apapun.

2.8 *Peak Signal to Noise Ratio* (PSNR)

PSNR atau *Peak Signal to Noise Ratio*, adalah istilah rekayasa untuk rasio antara daya maksimum yang mungkin dari sinyal dan kekuatan noise yang merusak mempengaruhi kesetiaan perwakilannya. Karena banyak sinyal memiliki *dynamic range* yang sangat luas, PSNR biasanya dinyatakan dalam skala logaritmik desibel.

PSNR adalah paling sering digunakan sebagai ukuran kualitas rekonstruksi codec kompresi lossy (misalnya, untuk kompresi gambar). Sinyal dalam hal ini adalah data asli, dan kebisingan adalah kesalahan diperkenalkan oleh kompresi. Ketika membandingkan codec kompresi digunakan sebagai pendekatan untuk persepsi manusia kualitas rekonstruksi, sehingga dalam beberapa kasus satu rekonstruksi mungkin tampak lebih dekat dengan aslinya daripada yang lain, meskipun memiliki PSNR bawah (PSNR yang lebih tinggi biasanya akan menunjukkan bahwa rekonstruksi adalah kualitas yang lebih tinggi). Salah satu harus sangat berhati-hati dengan berbagai validitas metrik ini, melainkan hanya konklusif berlaku bila digunakan untuk membandingkan hasil dari codec yang sama (atau jenis codec) dan konten yang sama.

Hal ini paling mudah didefinisikan melalui *mean squared error* (MSE) dimana untuk dua gambar monokrom m × n piksel, I dan K adalaha gambar monokrom yang dimana salah satu gambar yang dianggap sebagai gambar *noise* dari yang lain, dengan rumus :

$$MSE = \frac{1}{m \times n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i, j) - K(i, j)]^2$$

Dan PSNR sendiri didefinisikan sebagai :

$$PSNR = 10.\log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20.\log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right)$$

Dimana, MAX_I = nilai maksimum piksel