BAB 2

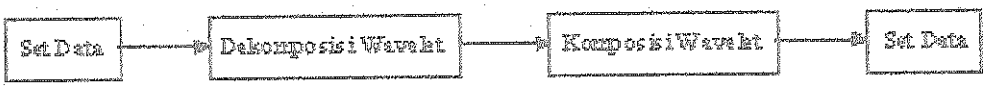
LANDASAN TEORI

2.1. Teori

2.1.1. *Haar Wavelet*

Untuk lebih mudah mengetahui apa itu *wavelet* dalam dunia matematika, sebagai permulaan kita ilustrasikan sebuah bola *billiard* yang bundar dan halus. Kita dapat membentuk bola *billiard* yang identik dengan seperangkat blok lego. Bila kita menggunakan blok lego yang besar maka akan terbentuk bola yang kasar, sebaliknya bila kita menggunakan blok lego yang kecil maka akan terbentuk bola yang lebih halus. *Wavelet* dalam ilustrasi di atas kira-kira memiliki peran seperti blok lego, di mana kita dapat memilih seberapa besar *wavelet* dengan memilih *scaling factor* tertentu, kita juga dapat melakukan pergeseran di sekitarnya dengan *displacement factor* sehingga bila dihubungkan dengan blok lego, mereka akan saling berkaitan dan membentuk sebuah bola *billiard*.

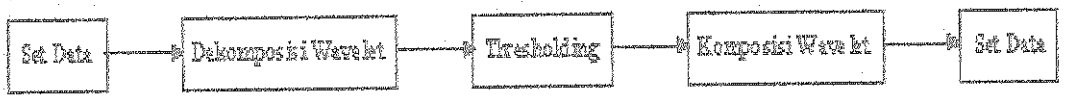
Ide dari *wavelet* itu sendiri secara umum digunakan sebagai tolak ukur *scaling and displacement factor* seperti yang telah diuraikan di atas : diberikan sebuah fungsi tertentu, kita mau memperoleh koefisien-koefisiennya, kita sebut fungsi itu '*wavelet*'. Tepatnya sekali kita mengetahui bagaimana menghitung *wavelet* dari sebuah fungsi atau set data tertentu, kita dapat mendapatkan kembali sebuah aproksimasi dari fungsi atau set data tersebut dengan membalik perhitungannya, untuk lebih jelasnya dapat diilustrasikan dengan bagan sebagai berikut :



Gambar 2.1 Kosep Dasar *Wavelet*

Dalam gambar 2.1, diberikan suatu set data, untuk set data berupa gambar biasanya datanya berupa nilai-nilai piksel dalam gambar tersebut, sehingga jumlah set datanya sebanyak luas dari gambar tersebut, yaitu lebar dikalikan dengan tinggi gambar tersebut dalam satuan piksel. Kemudian terhadap set data tersebut dilakukan dekomposisi *wavelet*, untuk mendapatkan detil gambar. Gambar yang telah didekomposisikan tersebut dapat direkonstruksi kembali, yaitu yang dinamakan dengan komposisi *wavelet*, sehingga didapatkan kembali aproksimasi dari set data sebelumnya.

Kekuatan *wavelet* yang sesungguhnya adalah kemampuannya untuk menghilangkan beberapa komponen dari vektor *wavelet*, tentunya tanpa memberikan pengaruh yang cukup besar saat aproksimasi set data berlangsung, umumnya ukuran *wavelet* identik dengan ukuran set data diskrit yang sedang dianalisis. Bagaimana pun, semuanya itu tergantung pada struktur dari *wavelet*nya itu sendiri, kadang kala mungkin untuk menghilangkan nilai hingga 80% dari *wavelet* dan tetap memperoleh aproksimasi set data yang cukup baik (contohnya, FBI menggunakan sampai 20:1 kompresi *wavelet* untuk memperkecil ukuran data sidik jari yang disimpan dalam basis data). Proses menghilangkan atau menyingkirkan koefisien tertentu dari suatu *wavelet* disebut dengan *thresholding*. Sehingga bagan di atas dapat dikembangkan menjadi seperti bagan berikut :



Gambar 2.2 Konsep Dasar *Wavelet* dengan *Thresholding*

Ide dari *wavelet* bukanlah hal yang baru : pada awal tahun 1800, Joseph *Fourier* menemukan bahwa sangatlah mungkin untuk melakukan superposisi *sines* dan *cosines* untuk merepresentasikan fungsi-fungsi lainnya. Hukumnya fungsi $f(x)$ dengan periodik 2π dapat dijabarkan sebagai berikut:

$$f(x) = a_0 + \sum_{k=1}^{\infty} (a_k \cos kx + b_k \sin kx)$$

di mana :

$$a_0 = \frac{1}{2\pi} \int_0^{2\pi} f(x)dx \quad , \quad a_k = \frac{1}{\pi} \int_0^{2\pi} f(x)\cos(kx)dx \quad , \quad b_k = \frac{1}{\pi} \int_0^{2\pi} f(x)\sin(kx)dx$$

untuk $k = 1,2,3,...$

Dengan kata lain , beberapa fungsi lainnya dapat diaprosimasikan melalui deretan gelombang *sines* dan *cosines*, di mana gelombang itu dapat diberlakukan pergeseran dan penyekalaan.

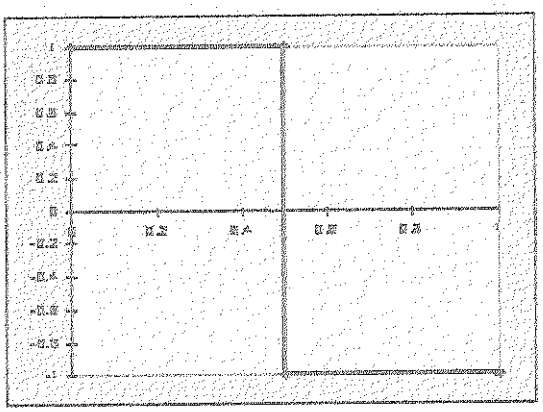
Masalah utama yang muncul dalam deret *Fourier* adalah mereka tidak terlokalisasi secara baik : jika kita ingin mempelajari sebuah fungsi dengan interval $[a,b]$, kita masih harus melibatkan semua koefisien *Fourier* (a_k, b_k). Dengan *wavelet* , kita hanya perlu memilih koefisien yang berhubungan dalam interval tertentu. Di

samping itu, koefisien *wavelet* lebih sederhana untuk dihitung dibandingkan dengan koefisien *Fourier* yang tertera di atas.

Wavelet Haar, yang ditemukan sekitar tahun 1909 didefinisikan sebagai berikut:

$$h(x) = \begin{cases} 1 & untuk \quad 0 < x \leq \frac{1}{2} \\ -1 & untuk \quad \frac{1}{2} < x \leq 1 \\ 0 & untuk \quad x \leq 0 \; atau \; 1 < x \end{cases}$$

Dalam representasi grafik, fungsi tersebut sangatlah sederhana :



Gambar 2.3 Grafik Fungsi *Wavelet Haar*

Wavelet Haar juga dikenal sebagai '*mother wavelet*' (atau sebuah *wavelet* induk yang telah digunakan sejak pertama kali secara intensif). Sangatlah penting untuk diperhatikan bahwa integral dari *wavelet Haar* di sepanjang domain adalah 0 (nol). Begitu juga integral dari fungsi *wavelet Haar* ketika diskalakan dengan suatu faktor juga memberikan hasil 0 (nol).

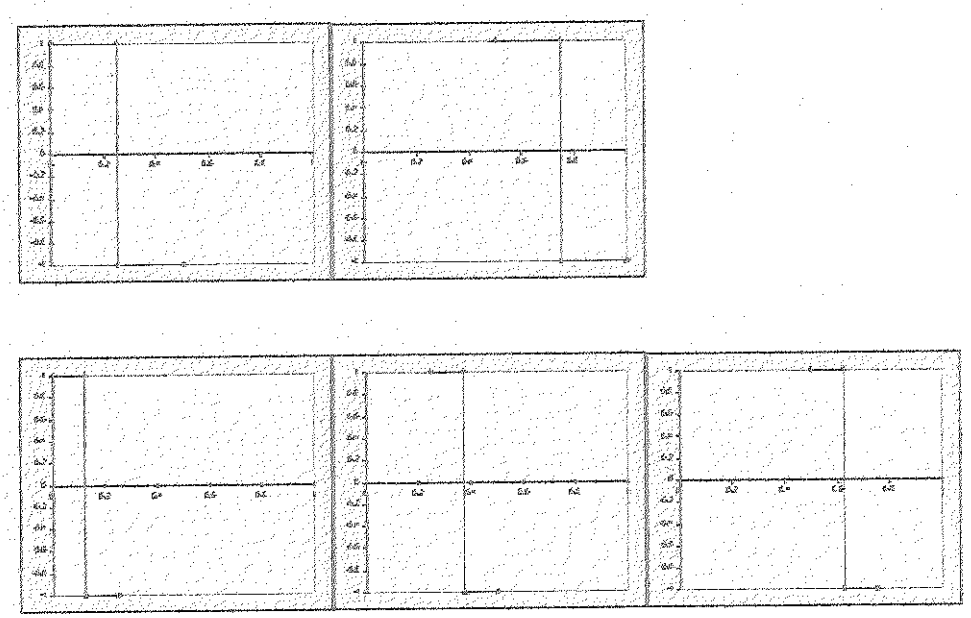
Untuk melakukan penyekalaan dan pergeseran *wavelet Haar* , didefinisikan fungsi *Haar* sebagai berikut :

$$h_{jk} = const \cdot h(2^j x - k), (j,k \in Z)$$

Untuk lebih jelasnya perhatikan grafik sebagai berikut (dengan urutan dari kiri ke kanan, atas ke bawah sebagai berikut : $h_{10}, h_{11}, h_{20}, h_{21}, h_{22}$). Di mana hal yang perlu diperhatikan :

(a) Untuk setiap nilai j yang diberikan , grafik tidak mengalami penimpaan.

(b) Grafik seluruhnya jatuh dalam '*power of 2 subdivision*' dari interval [0,1] (dengan kata lain, domainnya membentuk $[2^{-a}, 2^{-a+1}]$, di mana a adalah nilai *integer*). Jadi untuk kenaikan nilai j, kita mengetahui bahwa grafik fungsi akan jatuh seluruhnya 'di bawah' grafik fungsi dengan nilai j yang lebih kecil.



Gambar 2.4 Grafik Fungsi *Wavelet Haar* Perubahan Nilai j dan k

Fungsi h_{jk} tertulis di atas membentuk sebuah basis ortogonal untuk $L^2(R)$. Artinya adalah terdapat fungsi kuadrat yang dapat diintegralkan (*integrable*), yang nilai integralnya terbatas, dapat diekspresikan sebagai sebuah kombinasi linear dari sejumlah fungsi h_{jk}. Di bawah akan dibuktikan bahwa suatu basis memiliki sifat ortogonalitas, dan nilai konstanta fungsi h_{jk} dapat dihitung.

Yang akan dibuktikan adalah *inner product* $< h_{jk}, h_{j'k'} >$ adalah 0 (nol) asalkan syarat $j = j'$ dan $k = k'$ tidak terpenuhi secara bersamaan (dengan kata lain, dua fungsi yang berbeda dalam suatu basis adalah ortogonal). *Inner product* sepanjang basis tertentu didefinisikan sebagai integral dari perkalian 2 fungsi h_{jk} sebagai berikut :

$$< h_{jk}, h_{j'k'} >= \int h_{jk} \cdot h_{j'k'} = 0$$

Diasumsikan j berbeda dengan j' (di mana $j < j'$). Dengan melihat kembali fungsi yang telah tercantum di atas : karena j dan j' adalah nilai *integer*, maka grafik dari fungsi $h_{j'k'}$ akan jatuh seluruhnya di *power of 2 subdivision* dari interval $[0,1]$. Karena grafik h_{jk} 'lebih besar' dan jatuh dalam *power of 2 subdivision* dalam suatu unit interval, maka grafik $h_{j'k'}$ memenuhi salah satu ketentuan sebagai berikut :

(a) Termuat seluruhnya dalam suatu set di mana h_{jk} adalah 0 (nol), dalam hal ini integralnya adalah nol, karena hasil perkaliannya adalah 0 (nol).

(b) Dimuat seluruhnya dalam suatu set di mana h_{jk} adalah positif, dalam hal ini yang dilakukan hanyalah mengintegralkan versi skala dari $h_{j'k'}$ dengan suatu

konstanta, nilai integral tersebut bernilai 0 (nol), sebagaimana telah dibuktikan sebelumnya di atas.

(c) Dimuat seluruhnya dalam suatu set di mana h_{jk} adalah negatif, dengan alasan yang sama nilai integralnya juga 0 (nol).

Perhatikan bahwa analisis di atas tidak berhubungan dengan nilai k, sebab perubahan nilai k secara sederhana hanya berpengaruh pada pergeseran fungsi ke sebelah kanan atau kiri ke dalam *power of 2 subdivision* lainnya dari suatu unit interval.

Sekarang, jika $j = j'$ dan $k \neq k'$, ini berarti bahwa kedua fungsi sama hanya jatuh dalam interval yang terpisah, dengan teori yang telah dijelaskan sebelumnya, maka hasil kalinya juga 0 (nol).

Dengan analisis di atas maka terbukti bahwa *inner product* $< h_{jk}, h_{j'k'} >$ adalah 0 (nol) asalkan syarat $j = j'$ dan $k = k'$ tidak terpenuhi secara bersamaan.

Selanjutnya adalah adanya sifat *normality*, yang didefinisikan sebagai berikut:

$$< h_{jk}, h_{jk} >= \int h_{jk} \cdot h_{jk} = \int (h_{jk})^2 = 1$$

Dan untuk mencari konstantanya adalah dengan cara sebagai berikut :

$$1 = const^2 \int (2^j x - k)dx = const^2 \cdot 2^{-j} \int h(t)^2 dt = const^2 \cdot 2^{-j} \Rightarrow const = 2^{\frac{j}{2}}$$

Dengan hasil-hasil tersebut di atas, definisi formal dari basis *Haar* dari $L^2(R)$ menjadi :

$$h_{jk} = (2^{\frac{j}{2}} \cdot h(2^j x - k), untuk\ j,k \in Z\ dan\ x \in [0,1])$$

Sebuah fungsi f(x) dapat direpresentasikan dalam basis *Haar* sebagai berikut :

$$f(x) = \sum_{j,k} \langle f, h_{jk} \rangle h_{jk}(x) = \sum_{j,k} d_{jk} \cdot h_{jk}(x)$$

Koefisien d_{jk} disebut dengan the *Haar Wavelet coefficients* $\left(d_{jk} = \langle f, h_{jk} \rangle\right)$. Perhatikan pula bagaimana kita dapat 'memilih' untuk merepresentasikan fungsi tersebut dengan beragam 'detail' level :

- Pada detail level 1, fungsi di aproksimasikan 'secara kasar' dengan $f(x) = d_{00}h_{00} + d_{10}h_{10} + d_{11}h_{11}$.
- Pada detail level 2, fungsi diaproksimasikan dengan $f(x) = d_{00}h_{00} + d_{10}h_{10} + d_{11}h_{11} + d_{20}h_{20} + d_{21}h_{21} + d_{22}h_{22} + d_{23}h_{23}$.

Berikutnya diberikan definisi (*scaling function*) sebagai berikut :

$$\chi(x) = \begin{cases} 1, & 0 < x \le 1 \\ 0, & x \le 0\ atau\ x > 1 \end{cases}$$

$$\chi_{jk}(x) = 2^{\frac{j}{2}}\ \chi(2^j x - k)$$

Sangatlah mudah untuk memeriksa bahwa 'normal' dari χ_{jk} adalah 1 (satu), dan mulai saat ini digunakan faktor $2^{j/2}$ di depannya.

Karena bekerja dengan fungsi kontinu $f(x)$ sangat sulit, maka akan diambil suatu set nilai '*sample*' yang diambil dari fungsi, yang akan diterapkan dalam *Haar*

algorithm. Misalkan titik sampel itu dilambangkan dengan s_{jk} (titik-titik pada dasarnya menjadi nilai rata-rata dari fungsi yang diberikan pada interval yang bersebelahan) dedefinisikan sebagai berikut :

$$s_{jk} = \int f(x) \cdot \chi_{jk}(x)\, dx = \langle f, \chi_{jk} \rangle$$

Kita hanya melihat integral dari fungsi f sepanjang interval $[2^{-j}k, 2^{-j}(k+1)]$: hal ini dikarenakan fungsi χ_{jk} mempunyai nilai *non trivial* sepanjang interval tersebut. Perhatikan dengan seksama, bahwa integral tersebut memiliki skala menurun dengan faktor $2^{j/2}$ yang termuat dalam ekspresi χ_{jk}, yang berlaku tepatnya di sepanjang interval tersebut. *Scaling function* sesuai dengan namanya, memiliki peranan secara jelas bahwa fungsi $f(x)$ tersebut memberi pengaruh skala di sepanjang interval memberikan kita nilai rata-rata fungsi di sepanjang interval.

Untuk lebih memperjalas diberikan rangkuman sebagai berikut :

$d_{jk} = \langle f, h_{jk} \rangle$: koefisien *Haar*, dengan nilai h_{jk} didefinisikan :

$h_{jk} = (2^{\frac{j}{2}} \cdot h(2^j x - k), untuk\ j,k \in Z\ dan\ x \in [0,1])$

$s_{jk} = \langle f, \chi_{jk} \rangle$: nilai-nilai yang disampelkan dari $f(x)$,

$\chi_{jk}(x) = 2^{\frac{j}{2}}\, \chi(2^j x - k)$

Berikutnya kita akan menghitung $s_{j,2k} - s_{j,2k+1}$. Kuantiti ini disebut sebagai *difference* antara dua nilai sampel yang bersebelahan dari fungsi yang telah dianalisis sebelumnya, dengan diberikannya suatu 'detail' level (dalam kasus ini, level ' j '). Alasan mengapa kita melakukan ini, karena kita mau melihat apakah adanya relasi antara koefisien *Haar* dengan nilai-nilai sampel tersebut dari fungsi yang telah dianalisis.

Perlu diingat bahwa koefisien *Haar* adalah *inner product* antara fungsi 'aktual' dan fungsi *Haar*. Sebenarnya kita dapat membangun sebuah relasi antara koefisien *Haar* dengan nilai-nilai sampel tanpa memiliki fungsi yang dikerjakan. Dengan kata lain, diberikan suatu set nilai-nilai, kita dapat memperoleh koefisien *Haar* tanpa membutuhkan fungsi yang membangkitkan nilai-nilai tersebut. Di bawah ini adalah proses '*discretizing*' transformasi *wavelet* (kita sedang bekeja dengan suatu set nilai diskrit tanpa mempedulikan seluruh fungsi kontinunya).

$$\begin{aligned} s_{j,2k} - s_{j,2k+1} &= \int f \cdot \chi_{j,2k} - \int f \cdot \chi_{j,2k+1} = \int f \cdot (\chi_{j,2k} - \chi_{j,2k+1}) \\ &= \int f \cdot 2^{\frac{j}{2}} (\chi(2^j x - 2k) - \chi(2^j x - (2k+1))) \end{aligned}$$

Selanjutnya dihitung *difference* dari fungsi χ secara jelas :

$$\begin{aligned} &\chi(2^j x - 2k) - \chi(2^j x - (2k+1)) \\ &= \begin{cases} 1 - 0 = 1 & 2^j x - 2k \in (0,1] \\ 0 - 1 = -1 & 2^j x - (2k+1) \in (0,1] \end{cases} \\ &= \begin{cases} 1 & x \in (\frac{2k}{2^j}, \frac{2k+1}{2^j}] \\ -1 & x \in (\frac{2k+1}{2^j}, \frac{2k+2}{2^j}] \end{cases} \\ &= \begin{cases} 1 & x \in (2^{-(j-1)}k, \frac{2k+1}{2^j}] \\ -1 & x \in (\frac{2k+1}{2^j}, 2^{-(j-1)}(k+1)] \end{cases} \\ &= h(2^{j-1} x - k) \end{aligned}$$

Dan akhirnya :

$$s_{j,2k} - s_{j,2k+1} = \int f \cdot 2^{\frac{1}{2}} 2^{\frac{j-1}{2}} h(2^{j-1}x - k) = \sqrt{2} \int f \cdot h_{j-1,k} = \sqrt{2}\, d_{j-1,k}$$

Kesimpulannya , kita mendapatkan suatu hubungan :

$$\frac{1}{\sqrt{2}}(s_{j,2k} - s_{j,2k+1}) = d_{j-1,k}$$

Apakah maksud dari kesimpulan ini ? Artinya adalah sekali kita menghitung nilai-nilai sampel dari sebuah fungsi pada 'level berikutnya', kita mendapatkan nilai-nilai koefisien *Haar* pada 'level sebelumnya'. Salah satu cara untuk memperhatikannya adalah dengan $s_{j,k}$ sebagai waktu rataan dari fungsi f pada k^{th} interval waktu denngan panjang 2^{-j}; $d_{j,k}$ merepresentasikan variasi sinyal waktu rataan dari dua sinyal yang datang secara berurutan.

Relasi di atas membuahkan rekursif algoritma sebagai berikut :

- Dimulai dengan level suatu level j, dengan 2^j set nilai sampel.
- Hitung koefisien $d_{j-1,k}$ pada level $j-1$. Perhatikan, karena kita melakukan semuanya ini secara berpasangan menurut indeks ganjil-genap, maka jumlah dari keofisien d pada level $j-1$ akan menjadi 2^{j-1}.
- Hitung nilai-nilai sampel pada level $j-1$ (dengan cara menghitung rata-ratanya). Dan lagi kita akan memperoleh 2^{j-1} nilai-nilai sampel pada level $j-1$.

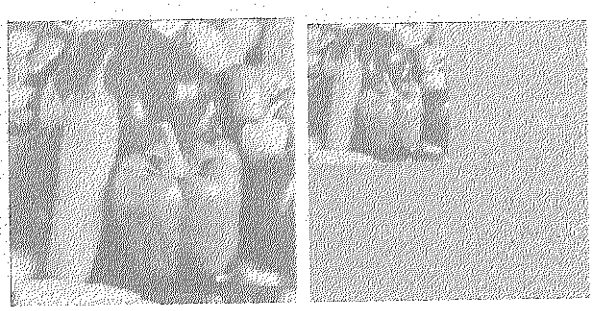
Lakukan secara rekursif langkah-langkah di atas.

2.1.2. *Discrete Wavelet Transform* (DWT)

Transformasi *wavelet* adalah transformasi matematika seperti transformasi *Fourier* yang sudah dikenal secara umum. Analisis *wavelet* adalah analisis multiresolusi, yang artinya koefisien *wavelet* untuk fungsi tertentu memuat keduanya domain frekuensi dan waktu. Kenyataan inilah yang membuat *wavelet* berdaya guna untuk aplikasi-aplikasi yang melibatkan pemrosesan sinyal (*signal processing*), di mana informasi dari frekuensi dan lokasi waktu dari suatu frekuensi sangat berguna.

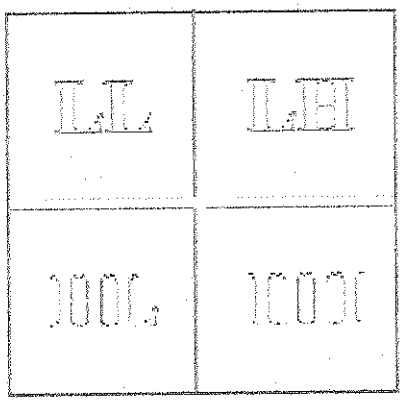
Dalam lingkup *image processing*, transformasi *wavelet* dilakukan dengan cara melakukan penyaringan koefisien gambar secara baris dan kolom. Setelah masing-masing transformasi selesai (satu kali transformasi berarti melakukan transformasi baris dan kolom secara berurutan masing-masing satu kali), koefisien *low pass* dari gambar akan ditransformasikan lagi. Proses ini dapat diulang biasanya sampai tinggi atau lebar area yang ditransformasikan (*low pass*) tidak lagi habis dibagi dua atau sama dengan satu.

Contoh dari transformasi *wavelet single pass* (satu *stage*) dapat digambarkan sebagai berikut :



Gambar 2.5
(kiri) Gambar Asli, (kanan) Gambar *Single Pass* Transformasi *Wavelet*

Gambar di sebelah kiri adalah gambar asli sebelum melaului proses transformasi *wavelet*. Gambar yang di sebelah kanan gambar yang sudah disaring dengan tranformasi *wavelet* sebanyak satu kali. Dan gambar yang sebelah kanan direpresentasikan dengan daerah sebagai berikut (L=*low*, H=*High*):



Gambar 2.6 Domain Transformasi *Wavelet*

Yang dimaksud dengan *low pass* data adalah data yang terletak di sebelah kiri atas dari gambar yang sudah ditransformasi. Sedangkan untuk data *high pass* nya adalah tiga daerah sisanya, di mana pada gambar tersebut komponennya hampir tidak tampak, sebab data tersebut sebagian besar memuat informasi lemah dari gambar.

Kegunaan *wavelet* dalam kompresi gambar adalah terbentang dari kenyataan bahwa transformasi *wavelet* dapat dengan jelas memisahkan informasi *high-pass* dan *low-pass* secara piksel demi piksel. Jika yang diinginkan adalah tingkat kompresi yang tinggi, maka kita hanya berorientasi pada *low pass* nya saja, semakin kecil ukurannya maka akan semakin tinggi tingkat kompresinya. Jika untuk masalah kualitas ketajaman gambar, maka koefisien *high pass* yang dibutuhkan untuk memperoleh kualitas gambar tertentu sesuai keinginan.

2.1.3. *Vector Quantization* (VQ)

Inti dari kompresi gambar adalah ide dari kuantisasi dan aproksimasi. Selama sebuah gambar dalam sebuah format digital, gambar dapat direpresentasikan dengan suatu nilai-nilai, kemudian nilai-nilai tersebut diberlakukan suatu transformasi, umumnya berupa DFT (*Discrete Fourier Transform*) atau DWT (*Discrete Wavelet Transform*). Hal ini akan menghasilkan nilai-nilai *real* , maka di sinilah dilakukan *quantization*, dengan tujuan untuk mendapatkan aproksimasi dari nilai-nilai kontinu ke nilai-nilai diskrit. Selama proses *quantization* setiap piksel dipetakan ke suatu nilai diskrit. Setiap nilai-nilai *integer* tersebut sudah ditentukan akan jatuh di dalam *range* yang telah ditentukan, di mana nilai-nilai dalam *range* tersebut melambangkan suatu warna.

Kompresi gambar dengan teknik *vector quantization* (VQ) telah diteliti selama bertahun-tahun. Saat ini, beberapa algoritmanya telah diimplementasikan dalam perangkat keras oleh beberapa *vendor chip* grafik, tetapi sayangnya dekompresi VQ masih sangat sulit dilakukan pada perangkat keras, dan lebih mudah bila dilakukan pada perangkat lunak.

Singkatnya, *quantization* adalah prosedur aproksimasi nilai-nilai kontinu dengan nilai-nilai diskrit. Tujuan dari *quantization* secara umum digunakan untuk menghasilkan representasi data yang lebih sederhana. Sebagai contoh, untuk menampung intensitas suatu warna kita dapat mengkuantisasikan nilai-nilai *real* dalam *range* [0.0, 1.0] ke nilai-nilai *integer* dalam *range* 0-255, merepresentasikannya dengan 8 *bit*, sehingga menyebabkan kebutuhan akan resolusi untuk beberapa aplikasi disesuaikan dengan warna. Contoh lainnya :

Gambar 2.7 Representasi Kuantisasi

Dari gambar di atas, setiap angka yang kurang dari -2 diaproksimasikan dengan angka -3. Setiap angka di antara -2 dan 0 diaproksimasikan dengan angka -1. Setiap angka di antara 0 dan 2 diaproksimasikan dengan angka 1. Setiap angka lebih besar dari 2 diaprosimasikan dengan angka 3. Perlu diperhatikan pula nilai-nilai aproksimasi secara unik direpresentasikan sebanyak 2 *bit*.

Contoh di atas adalah contoh kuantisasi skalar, di mana nilai input dan nilai outputnya berupa nilai skalar, atau berupa nilai tunggal. VQ adalah perluasan dari *scalar quantization* dengan bertumpu pada sebuah vektor. Struktur dasar dari VQ pada dasarnya mirip dengan *scalar quantization*, dan juga memuat *encoder* dan *decoder* pula. *Encoder* menentukan ruang partisi input vektor, dan pada setiap partisi ditentukan sebuah faktor atau indeksnya, yang dikenal sebagai *codeword*, sejumlah set dari *codeword* disebut dengan *codebook*. Sedangkan *decoder* malakukan pemetaan tiap-tiap indeks untuk menghasilkan sebuah vektor.

Kuantisasi yang digunakan dalam perancangan ini digunakan metode *uniform vector quantization*, yaitu metode kuantisasi yang paling sederhana. Secara sederhana nilai hasil kuantisasi dapat dijabarkan sebagai berikut :

$$nilai\ kuantisasi = round(nilai\ koefisien\ real \quad x \quad fungsi\ kuantisasi)$$

Sedangkan proses *dequantization* adalah proses untuk mendapatkan kembali nilai-nilai koefisien *real* yang sebelumnya telah dikuantisasikan. Cara mencarinya :

nilai koefisien real = nilai aproksimasi x invers fungsi kuantisasi

2.1.4. *Embedded Zerotree Wavelet* (EZW)

Karena adanya struktur spesial dan sifat yang beragam dalam sebuah gambar yang direpresentasikan dengan koefisien DWT, sehingga metode *coding* yang khusus dapat dibentuk. Contohnya adalah sebagai berikut :

- Algoritma *Embedded Zerotree Wavelet* (EZW).
- Algoritma *Set Partitioning in Hierarchical Trees* (SPIHT).
- Algoritma *Embedded Block Coding with Optimized Truncation* (EBCOT), yang diadopsi ke dalam kompresi gambar standar JPEG2000.

Tujuan dari algoritma-algoritma di atas adalah sama, yaitu untuk membentuk suatu prosedur pengkodean gambar yang efektif dan perhitungan yang efisien. Diberikan suatu nilai *threshold* T, sebuah koefisien x adalah sebuah elemen dari *zerotree* signifikan adanya nilai *threshold* sebagai tolak ukurnya. Kesignifikanan dalam *zerotree* terdiri dari empat simbol sebagai berikut :

- *The Zerotree Root*, suatu root dari sebuah *zerotree* dikodekan dengan simbol tertentu yang mengindikasikan bahwa telah diprediksikannya koefisien-koefisien pengikutnya sedikit yang signifikan nol.

- *Isolated zero*, suatu koefisien adalah signifikan nol tetapi memiliki banyak pengikut yang signifikan nol.
- *Positive Significance*, suatu koefisien adalah signifikan dengan nilai positif.
- *Negative Significance*, suatu koefisien adalah signifikan dengan nilai negatif.

EZW terdiri dari dua macam proses, yaitu *encoding* dan *decoding*. Proses *encoding* adalah berupa proses pengkodean dengan memberikan simbol-simbol signifikan dari suatu nilai-nilai koefisien DWT yang sudah melalui proses *quantization*. Sedangkan *decoding* adalah berupa proses pemecahan keempat macam simbol-simbol signifikan tersebut yang telah ditentukan dalam proses *encoding*, sehingga untuk tahap selanjutnya dapat dilakukan invers *quantization* dan invers *taransform coding*. Proses *decoding* disebut juga sebagai proses rekonstruksi.

2.1.5. *Run-Length Encoding* (RLE)

RLE adalah suatu *entropy encoding* yang sangat sederhana untuk kompresi data. Cara kerja algoritma ini dalam kompresi data adalah dengan menelusuri aliran *bit* yang memiliki nilai yang sama untuk dikodekan dengan satuan *bit* saja. Kata *run* berarti dalam algoritma ini dilakukan pengkodean secara langsung dari aliran-aliran *bit* yang akan dikompresi.

Algoritma RLE biasanya digunakan pada beberapa data dengan memperhatikan isi dari data tersebut, sehingga baik dan buruknya kompresi suatu data ditentukan juga oleh isi dari data yang akan dikompresi. Kompresi RLE biasanya akan semakin baik

untuk data teks yang banyak memiliki *multiple space* atau pada data gambar *bitmap* yang memiliki jumlah koefisien barisan nol.

RLE adalah salah satu algoritma kompresi *lossless* yang sangat sederhana dan mudah diimplementasikan. Contohnya bila kita memiliki barisan nilai 3 4 0 0 0 0 3 2 1, maka dapat dikodekan dengan algoritma RLE sehingga menjadi 3 4 0 * 4 3 2 1.

2.1.6. *Lossy Compression*

Bila kita melakukan kompresi suatu data, setelah itu data tersebut di dekompresi kembali, tetapi hasil dekompresinya tidak sama dengan data aslinya, tetapi "mendekati" aslinya, inilah yang dimaksud dengan kompresi data *lossy*, yang tentunya memiliki banyak kegunaan. Tipe kompresi ini sangat sering digunakan dalam dunia internet, khususnya dalam *streaming media* dan aplikasi *telephony*, metode ini secara khusus disebut dengan *codecs*.

Ruang lingkup kompresi data *lossy* terbagi menjadi 2 bagian, yaitu :

- *Lossy transform codecs*, mula-mula diambilnya sampel dari gambar atau suara, kemudian datanya dipilah-pilah menjadi bagian yang kecil, kemudian dilakukan transformasi ke ruang basis yang baru, dan dikuantisasikan. Nilai-nilai yang diperoleh dari kuantisasi dilanjutkan dengan pengkodean *entropy*.
- *Lossy predictive codecs*, data dekode digunakan untuk memprediksi sampel suara atau gambar yang dikompresi. Kesalahan antara data yang diprediksi dan data yang sesungguhnya, secara bersama-sama dengan tambahan informasi yang dibutuhkan dilakukan kuantisasi.

Keuntungan kompresi data *lossy* dibandingkan dengan kompresi data *lossless* (metode kompresi yang hasil dekompresinya sama dengan data aslinya) adalah dalam banyak permasalahan metode kompresi *lossy* dapat menghasilkan *file* terkompresi yang jauh lebih kecil bila dibandingkan dengan menggunakan metode kompresi *lossless*. Oleh sebab itu metode kompresi *lossy* sangat cocok digunakan untuk kompresi data gambar atau suara.

2.1.7. *Wavelet-based Image compression*

Kompresi gambar dengan menggunakan *wavelet* terutama ditujukan untuk kompresi gambar *lossy*. Langkah-langkahnya adalah sebagai berikut :

- Digitalisasikan gambar sumber ke dalam signal s, hasil sinyalnya berupa nilai-nilai *integer*.
- Dekomposiskan sinyal-sinyal tersebut ke dalam urutan koefisien *wavelet* w.
- Gunakan *thresholding* untuk memodifikasi koefisien *wavelet* dari w ke w'.
- Gunakan kuantisasi untuk mengkonversi w' ke q.
- Aplikasikan pengkodean *entropy* untuk kompresi q ke e.

Langkah pertama dalam proses kompresi *wavelet* adalah digitalisasi sebuah gambar. Digitalisasi gambar dapat direpresentasikan dengan menggunakan tingkat keabuan (*gray scale level*) dengan *range* dari 0 (hitam) sampai 255 (putih).

Pada beberapa sinyal, beberapa koefisien *wavelet* mendekati nilai 0 (nol). Melalui metode yang dinamakan *thresholding*, koefisien-koefisien ini dinolkan, sehingga sering dijumpai banyaknya urutan nol dalam koefisien-koefisien tersebut.

Melalui tipe kompresi yang dikenal dengan nama pengkodean *entropy*, nilai-nilai koefisien nol yang berurutan dapat disimpan secara digital tanpa memakan banyak tempat.

Tahap keempat dari proses di atas adalah kuantisasi, seperti yang telah disebutkan dalam subbab sebelumnya, dapat diartikan mengubah urutan nilai *real* w' ke dalam urutan nilai *integer* q. Bentuk yang paling sederhana adalah dengan melakukan pembulatan ke nilai *integer* terdekat.

2.1.8. Gambar Digital

Gambar sebagai keluaran suatu sistem perekaman data dapat bersifat optik berupa foto, bersifat analog berupa sinyal-sinyal video seperti gambar pada monitor televisi, atau bersifat digital yang dapat langsung disimpan dalam pita magnetik.

Komputer hanya dapat bekerja dengan bilangan numerik yang berhingga, sehingga gambar harus diubah ke dalam bentuk bilangan numerik berhingga (gambar digital) sebelum diproses dalam suatu komputer. Untuk mengubah gambar yang bersifat kontinu menjadi gambar digital, diperlukan proses pembuatan kisi-kisi arah horisontal dan vertikal, sehingga diperoleh gambar dalam bentuk *array* dua dimensi. Proses tersebut dikenal sebagai proses digitalisasi atau sampling.

Gambar dapat dipresentasikan dengan *array* dua dimensi, di mana setiap elemen *array* dikenal sebagai elemen gambar (*picture elemen / pixel*) atau piksel. Pembagian suatu gambar menjadi sejumlah piksel dengan ukuran tertentu ini akan menentukan resolusi *spatial* (derajat kehalusan) yang diperoleh. Semakin kecil ukuran pikselnya, maka akan semakin halus gambar yang diperoleh, karena informasi yang hilang akibat

pengelompokan tingkat keabuan pada proses pembuatan kisi-kisi akan semakin kecil. Angka pada gambar digital menggambarkan bagaimana setiap piksel menggambarkan kecerahan (*brightness*) gambar tersebut pada titik yang bersangkutan.

Proses lain yang dapat dilakukan dalam suatu gambar digital adalah proses kuantisasi (*quantization*). Dalam proses ini tingkat keabuan setiap piksel dinyatakan sebagai suatu harga *integer*. Batas-batas harga *integer* atau besarnya daeerah tingkat keabuan yang digunakan untuk menyatakan tingkat keabuan piksel akan menentukan resolusi kecerahan dari gambar yang diperoleh. Kalau digunakan 3 *bit* untuk menyimpan harga *integer* tersebut, maka akan diperoleh sebanyak 8 tingkat keabuan.

Seluruh tahapan proses konversi di atas dikenal sebagai konversi analog ke digital, yang biasanya akan menyimpan hasil proses pada memori gambar. Sebaliknya, sebagai hasil suatu proses pengolahan gambar digital, kadang-kadang perlu mengeluarkan gambar dari memori gambar ke bentuk peragaan pada monitor televisi atau ke bentuk cetak foto. Proses konversi balikan ini dikenal sebagai konversi digital ke analog.

2.1.9. *The RGB(Red Green Blue) Color Model*

Pada dasarnya ada tiga warna utama untuk menampilkan grafik, yaitu warna merah, hijau, dan biru. Tentunya pewarnaan ini dilakukan oleh monitor dengan bantuan kartu grafik, yang melakukan kombinasi warna merah, hijau, dan biru dalam tiap pixel untuk memberikan 256 x 256 x 256 (24 *bit*) kombinasi warna yang berbada.

Pewarnaan RGB masing-masing dari warna utama memiliki nilai sebanyak 256 (0-255). Pewarnaan RGB dalam penggunaannya dapat direpresentasikan dalam bentuk 16 warna (0-15), di mana 0 (nol) mewakili warna hitam dan 15 mewakili warna putih.

Bisa juga direpresentasikan dalam bilangan heksadesimal dengan *range* *$000000* - *$FFFFFF*. Masing-masing dikelompokkan menjadi 1 *byte* (8 *bit*), secara berurutan dari kiri ke kanan mewakili warna merah, hijau, dan biru.

2.1.10. *YUV Color Model*

YUV adalah sistem pewarnaan yang digunakan dalam sistem warna televisi PAL, yang mulanya menjadi standar di Eropa. Y berarti komponen *luminance* (*brightness*), U dan V adalah komponen *chrominance*(*color*). Sistem pewarnaan YcbCr atau YpbPr, yang digunakan dalam komponen video komputer, adalah turunan dari pewarnaan YUV (Cb/Pb dan Cr/Pr adalah versi penyekalaan dari U danV), yang nantinya akan berhubungan dengan sistem pewarnaan YIQ untuk sistem pewarnaan televisi NTSC.

Sinyal YUV dibuat dari sumber warna RGB. Bobot dari R, G, dan B dijumlahkan seluruhnya untuk mendapatkan sinyal Y, yang merepresentasikan *brightness* keseluruhan, atau *luminance* dari suatu piksel. Sinyal U dibuat dengan mengurangi sinyal Y dari sinyal B dari RGB, dan dilakukan penyekalaan. Sedangkan sinyal V didapatkan dengan mengurangi sinyal Y dari sinyal R dari RGB dan dilakukan penyekalaan dengan faktor yang berbeda.

Sistem pewarnaan YUV dari RGB dapat dirumuskan sebagai berikut :

$$
\begin{aligned}
Y &= +0.299R + 0.587G + 0.114B \\
U &= +0.492\,(B - Y) \\
&= -0.147R - 0.289G + 0.436B \\
V &= +0.877\,(R - Y) \\
&= +0.615R - 0.515G - 0.100B
\end{aligned}
$$

Sedangkan untuk mendapatkan kembali sinyal RGB dari sinyal YUV yang ada dapat dirumuskan sebagai berikut :

$$R = Y + 1.140251V$$
$$B = Y + 2.028398U$$
$$G = (Y - 0.3R - 0.11B) \times 1.694915$$

2.1.11. *True Colour*

True Colour adalah suatu cara merepresentasikan warna dalam bilangan, baik berupa bilangan binari, desimal, atau heksadesimal, dan biasanya warna-warna tersebut dikodekan dari kombinasi tiga warna utama merah, hijau, dan biru. Jadi dapat dikatakan bila suatu piksel direpresentasikan sebanyak 3 byte (24 *bit*), terdapat 256 kemungkinan warna merah, 256 kemungkinan warna hijau, dan 256 kemungkinan warna biru. Bila direpresentasikan dengan 24 *bit*, maka akan terdapat 2^{24}(16.777.216) jumlah warna.

Representasi *true colour* yang lain dapat dilihat dalam tabel sebagai berikut di bawah :

Tabel 2.1 Tabel *True Colour*

Kedalaman warna	Jumlah warna maksimum	Ukuran *file* BMP(640x480)	
		Data *Bitmap*	*Header* dan *Footer*
1 *bit*	2	37,50 KB	0,96 KB
4 *bit*	16	150,00 KB	3,72 KB
8 *bit*	256	300,00 KB	8,28 KB
16 *bit*	65.536	600,00 KB	15,5 KB
24 *bit*	16.777.216	900,00 KB	21,65 KB

2.1.12. *Borland Delphi*

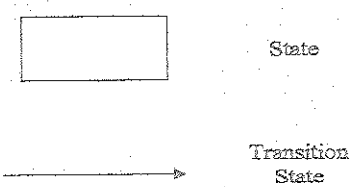
Borland Delphi merupakan piranti lunak pemrograman yang mempunyai cakupan kemampuan yang luas dan cukup canggih. Beragam aplikasi dapat dibuat dengan *Delphi*, di antaranya aplikasi pengolah teks, grafik, angka, basis data, aplikasi

web, dan yang terbaru aplikasi yang dapat diintegrasi dengan teknologi *.NET* dari *Microsoft*. *Delphi* menyediakan fasilitas pemrograman yang sangat lengkap, berbagai macam objek telah tersedia, dari objek yang berguna untuk aplikasi pengolah teks, gambar, video, hingga basis data. Mengingat adanya teknologi objek dan kelas, maka konsep *inheritance* atau penurunan sifat suatu objek sangat penting peranannya.

Keunggulan *Delphi* lainnya, adalah dengan diciptakan dua macam basis program untuk sistem operasi yang berbeda, yaitu VCL untuk sistem operasi *Microsoft Windows* dan CLX untuk sistem operasi dengan *Linux*. Yang lebih tangguh lagi, yaitu dengan didukungnya *run-time library* yang cukup lengkap. Dengan dilengkapinya *run-time library*, maka akan memudahkan sejumlah programmer dalam membuat suatu objek-objek dalam program aplikasi, sebab programmer tidak perlu lagi memikirkan sejumlah objek-objek komponen yang harus diletakkan di suatu tempat di atas suatu *form*.

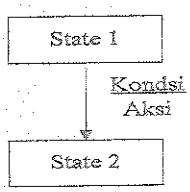
2.1.13. *State Transition Diagram* (STD)

STD merupakan *modelling tool* yang menggambarkan sifat ketergantungan waktu pada suatu sistem atau program. Pada mulanya STD hanya digunakan untuk menggambarkan suatu sistem atau program yang memiliki sifat *real-time*. Notasi-notasi STD adalah sebagai berikut :



Gambar 2.8 Notasi STD

State adalah kumpulan keadaan atau atribut yang mencirikan suatu benda atau keadaan tertentu. Perhatikan juga hubungan antar *state* berikut ini :



Gambar 2.9 Hubungan Antar *State*

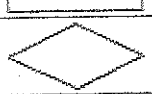
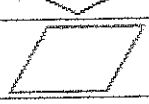
Kondisi adalah suatu kejadian yang dapat dideteksi oleh suatu sistem atau program. Sedangkan aksi dilakukan sistem atau program bila terjadi perubahan.

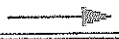
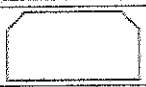
2.1.14. *Flowchart* (Diagram Alir)

Flowchart mengilustrasikan langkah-langkah dalam suatu proses. Dengan melihat proses-proses dalam *flowchart*, maka dapat dengan cepatnya membantu mengidentifikasikan *bottlenecks* atau tidak efisiennya suatu urutan logika, dan dapat diperbaiki dan dikembangkan dengan segera.

Notasi-notasi yang digunakan dalam *flowchart* adalah :

Tabel 2.2 Notasi *Flowchart*

Notasi	Keterangan
	*Start* \ *End*
	*Action* \ *Process*
	*Decision*
	*Input* \ *Output*

	*Flow Line*
	*Subroutine*
	*Loop Limit*

2.1.15. Interaksi Manusia Dan Komputer

Dalam merancang suatu program atau sistem, harus diperhatikan satu hal yang sangat penting, yaitu interaksi antara pengguna dengan program atau sistem. Interaksi ini harus *user friendly*, yang berarti mudah untuk digunakan oleh pengguna yang awam. Dalam meracang program atau sistem yang *user friendly* ada beberapa kriteria yang harus diperhatikan, yaitu :

- Waktu belajar singkat, sehingga pengguna yang awam dapat menggunakannya tanpa harus belajar dengan waktu yang lama.
- Kecepatan penyajian data atau informasi yang relatif cepat agar pengguna tidak menunggu terlalu lama dan menjadi bosan.
- Tingkat kesalahan yang dilakukan pengguna rendah.
- Penghafalan setelah melampaui jangka waktu tertentu, artinya seberapa jauh pengetahuan pengguna setelah sekian lama bila dihubungkan dengan frekuensi pemakaian.
- Kepuasan pribadi dari pengguna terhadap berbagai aspek yang terdapat dalam program atau sistem tersebut.

2.2. Penelitian yang Relevan

2.2.1. *FBI Fingerprint Compression*

Antara tahun 1924 dan 2002, US *Federal Bureau Investigation* telah memiliki sekitar 30 juta set sidik jari. Pada tahun 1993, *FBI's Criminal Justice Information Service Division* mengembangkan suatu standar untuk digitalisasi dan kompresi sidik jari bekerja sama dengan *National Institute of Standards and Technology, Los Alamos National Laboratory*, para vendor, dan komunitas penegak keadilan.

Kompresi gambar yang dipakai adalah kompresi gambar dengan penerapan teori *Wavelet*. Pada saat itu kompresi gambar dibutuhkan dengan alasan masalah penyimpanan data, khususnya kapasitas penyimpanan yang terbatas.

Gambar-gambar sidik jari melalui proses digitalisasi dan dihasilkan kembali dengan resolusi gambar 500 ppi (*pixels per inch*) dengan 256 tingkat keabuan informasi per pikselnya. Suatu gambar sidik jari disimpan sekitar 700.000 piksel dan membutuhkan penyimpanan sekitar 0,6 Mbytes (*megabytes*). Sedangkan untuk menyimpan seluruh sidik jari, jari-jari pada tangan kanan dan kiri, dibutuhkan penyimpanan sekitar 6 Mbytes. Jadi bila untuk menyimpan kesepuluh sidik jari, misalkan untuk 30 juta individu, dibutuhkan penyimpanan data sebesar kurang lebih 200 *terabytes*. Mengingat akan semakin banyaknya jumlah sidik jari yang akan disimpan, maka kompresi data, dalam hal ini data berupa gambar sangatlah dibutuhkan, selain untuk menghemat ruang penyimpanan, juga untuk menghemat biaya, belum lagi dibutuhkan penyimpanan untuk *backup* datanya.

Kompresi gambar sidik jari dengan *wavelet* yang diterapkan oleh FBI mampu sampai dengan perbandingan 1:26, di mana hasil kompresinya tidak jauh berbeda dengan gambar sebelumnya. Berikut adalah contoh gambar sidik jari untuk ibu jari

tangan kiri. Gambar di sebelah kiri adalah gambar aslinya, dan gambar di sebelah kanan adalah gambar hasil kompresinya.

Gambar 2.10
(kiri) Gambar sidik Jari Asli. (kanan) Gambar Sidik Jari Yang Terkompresi

2.2.2. *Light Field Compression* dengan Transformasi *Wavelet* dan Vektor Kuantisasi

Penelitian ini telah dilakukan oleh Li- Yi Wei pada tahun 1997. Menurut Beliau transformasi *wavelet* diikuti dengan vektor kuantisasi sangatlah cocok untuk melakukan kompresi *light field*. Alasannya, dengan melakukan transformasi *wavelet* kita dapat mendekomposisikan *light field* ke dalam *band* (daerah dengan range tertentu) yang terpisah, dan mengkompres masing-masing *band* menggunakan *bit-rate* yang berbeda berdasarkan seberapa penting peran suatu koefisien dalam suatu *subband*. Sedangkan vektor kuantisasi juga tidak kalah pentingnya, mengingat bahwa vektor kuantisasi memiliki sifat *encoding* dan *decoding* secara *real-time*.