BAB II

TINJAUAN PUSTAKA

2.1. Pengenal Tanda Tangan

Dalam penelitian yang dilakukan oleh Lau (1999) di *Hong Kong Baptist Univeristy,* tanda tangan memiliki arti dari identitas personal pada banyak negara dan banyak digunakan seperti pada *bank check,* kartu kredit, dan akses kontrol keamanan. Parizeau (1990) menyebutkan bahwa tanda tangan adalah cara terbaik untuk mengidentifikasi individu dengan tulisan tangan. Secara tradisional, tanda tangan dikenali oleh manusia.

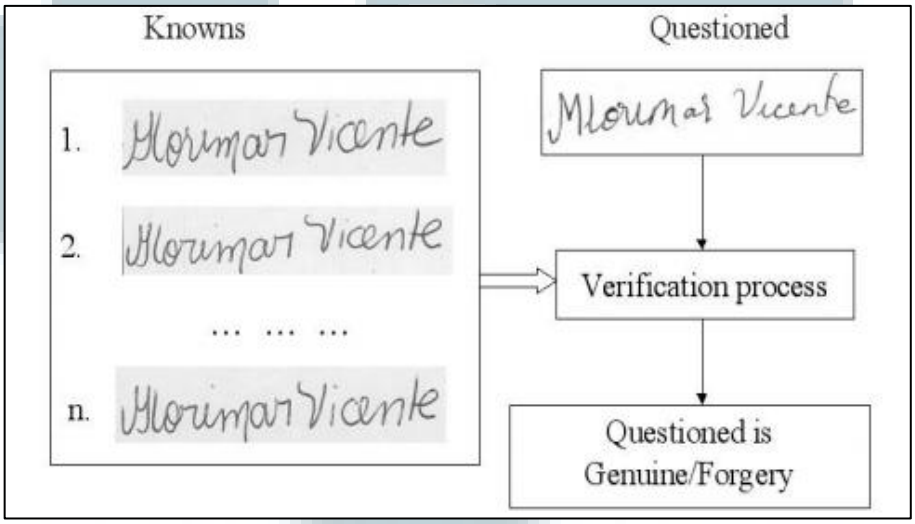
Menurut Lau (1999), aspek fisik dan psikologis menjadi alasan mengapa tanda tangan setiap orang akan sedikit berbeda. Variasi di dalamnya yakni

1. Berbeda translasi, rotasi, dan ukuran.
2. Satu garis dapat terbagi menjadi dua garis.
3. Dua garis tergabung menjadi satu garis.
4. Beberapa variasi garis.

Das (2004) menulis sebuah artikel mengenai pengenalan tanda tangan. Disebutkan bahwa teknologi pengenalan tanda tangan mengikutsertakan penggunaan sebuah pena dan perangkat tablet tulis khusus yang terhubung dengan komputer pemrosesan verifikasi yang lokal ataupun terpusat. Untuk pengambilan data tanda tangan, setiap individu diharuskan untuk melakukan tanda tangan beberapa kali pada perangkat tablet tulis. Harus diperhatikan bahwa kekuatan dari sebuah pengenalan tanda tangan tergantung pada kualitas perangkat tablet tulis. Perangkat berkualitas akan dapat menangkap segala aspek yang diperlukan seperti

waktu, tekanan, dan kecepatan. Bertolak belakang dengan perangkat tablet tulis yang berkualitas rendah yang mungkin tidak berhasil mendapatkan data tersebut di atas.

Menurut Ismail (2010), tujuan dari pengenalan tanda tangan adalah untuk mengenali penanda tangan untuk tujuan pengenalan dan verifikasi. Pengenalan adalah menemukan identifikasi dari pemilik tanda tangan. Verifikasi merupakan keputusan mengenai apakah tanda tangan tersebut asli atau palsu.



Gambar 2.1. Proses verifikasi tanda tangan
Sumber: Ismail (2010)

Ismail (2010) juga mengatakan bahwa pengenalan tanda tangan dapat dibagi menjadi dua macam yakni *on-line* (dinamis) dan *off-line* (statis). Pengenalan *on-line* mengacu pada proses dimana pengguna menggunakan pena spesial yang disebut dengan *stylus* untuk membuat tanda tangannya, hal ini menghasilkan lokasi pena, kecepatan, dan tekanan, sedangkan pengenalan *off-line* bekerja dengan gambar tanda tangan yang didapat dengan menggunakan pemindai ataupun kamera digital. Secara umum, pengenalan *off-line* adalah permasalahan

yang menantang. Pengenalan secara *off-line* lebih sulit jika dibandingkan dengan pengenalan *on-line*, dimana tidak terdapat informasi mengenai durasi, waktu pengorderan, jumlah goresan, dan arah penulisan. Tapi sistem *off-line* memiliki keuntungan yang signifikan yakni tidak diperlukannya sebuah perangkat yang spesial untuk menghasilkan tanda tangan tersebut.

2.2. Normalisasi

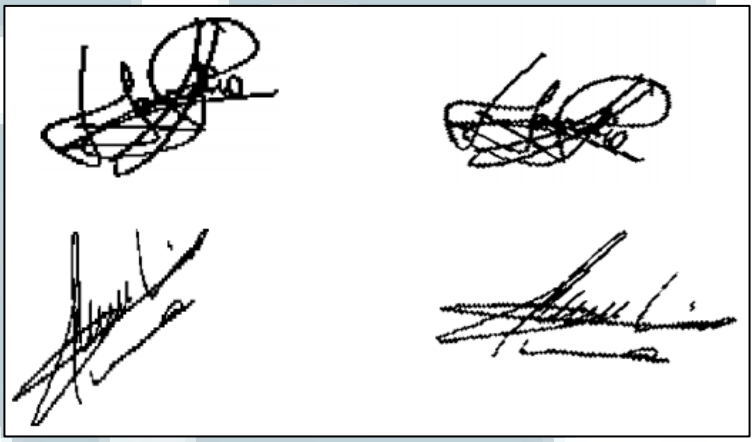
Kozielski (tanpa tahun) mengatakan bahwa teks dalam bentuk gambar tulisan tangan menampilkan perbedaan yang kuat dalam penampilannya dikarenakan perbedaan cara menulis. Perbedaan tampilan tersebut berada pada sisi ukuran kata, *slant, skew*, dan ketebalan garis. Perbedaan-perbedaan tersebut memicu pengembangan dalam normalisasi dan teknik *preprocessing* yang cocok untuk pengenalan teks tulisan tangan.

Menurut Kozielski (tanpa tahun) diantara langkah *preprocessing* yang paling umum, sistem *state-of-art* yang diterapkan antara lain *noise removal, binarization, skew* dan *slant correction, thinning,* dan *baseline normalization.*

Adler (1998) mengatakan metode tertentu yang digunakan untuk menangani *invariance* dalam pemrosesan gambar disebut sebagai *image normalization.* Menurut Adler (1998), secara intuitif, normalisasi adalah metode sistematis sederhana untuk merubah dari *observer-based* menjadi *image-based coordinate.*

Ismail (2010) mengatakan normalisasi sangatlah penting untuk menghasilkan *scaling* dan *rotation invariant* dari gambar target sebelum melakukan fase *recognition* atau otentikasi. Menurut Ismail (2010), *scale*

normalization dapat dilakukan dengan melakukan *scaling* pada koordinat x dan koordinat y secara berturut-turut ke ukuran yang sudah ditentukan sebelumnya. Untuk menormalisasi kemiringan (*slant*), Ismail (2010) mengusulkan algoritma *moment based*. Ide dasar dari algorintma ini, menurut Ismail (2010), adalah menghitung sudut kemiringan (*slant angle*) dari tulisan tangan atau tanda tangan merujuk pada *second moments* dari *foreground pixels* dan merotasi *foreground pixels* tersebut ke arah yang berlawanan.



Gambar 2.2. Hasil dari proses *slant normalization*
Sumber: Ismail (2010)

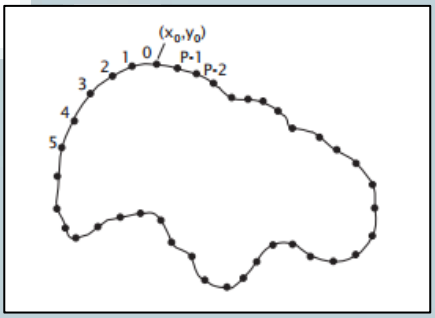
Gambar 2.2 bagian kiri merupakan gambar asli yang belum diproses dengan menggunakan metode *slant normalization* sedangkan gambar di sebelah kanan merupakan gambar tanda tangan hasil proses *slant normalization.*

2.3. Fourier Descriptor

Menurut Jähne (2005, pp. 522), *Fourier Descriptor* terbagi menjadi beberapa hal yakni.

1. *Cartesian Fourier Descriptor*

Fourier descriptor, mirip dengan kode rantai karena hanya menggunakan batas dari sebuah objek. Meskipun mirip, *fourier descriptor* juga memiliki perbedaan dengan kode rantai karena *fourier descriptor* tidak menjelaskan lengkungan pada *discrete grid*. Hal ini dapat diformulasikan untuk kurva sampel. Batas lengkung dapat diambil dengan panjang p dari titik awal $[x_0, y_0]^T$ sebagai parameter.



Gambar 2.3. Ilustrasi batas lengkung
Sumber: Jahne (2005, pp. 523)

Menurut Jahne (2005) tidak mudah untuk mendapatkan batas lengkung dengan sampel yang sama jauh. *Discrete boundary Curves*, seperti kode rantai, memiliki banyak kerugian. Dalam *8-neighborhood* sampel tidak sama jauh. Dalam *4-neighborhood*, sampel sama jauh, tetapi batasan bergerigi dikarenakan tiap bagian dari batas lengkung hanya dapat mengarah secara horizontal atau vertikal. Oleh karena itu, batas pinggir cenderung menjadi lebih panjang. Konsekuensinya, bukan menjadi ide yang baik untuk membentuk sebuah *continuous boundary curve* dari titik-titik pada *grid* biasa. Salah satu alternatifnya adalah melakukan *extract subpixel-accurate object boundary curves* secara langsung dari gambar *grayscale.*

Contiuous boundary curve adalah dari bentuk x(p) dan y(p). Kedua kurva ini dapat dikombinasikan dengan sebuah fungsi z(p) = x(p) + iy(p). Jika P adalah batas pinggir dari kurva, maka

$$z(p + nP) = z(p) \quad n \in \mathbb{Z}$$ ..... Rumus 2.1

Kurva periodik dapat diperluas dalam *fourier series.* Koefisien dari *fourier series* ini dipaparkan sebagai berikut

$$\hat{z}_v = \frac{1}{P}\int_0^P z(p)\exp\left(\frac{-2\pi i v p}{P}\right) dp \quad v \in \mathbb{Z}$$ ..... Rumus 2.2

Kurva periodik dapat direkonstruksi dari koefisien *Courier* dengan

$$z(p) = \sum_{v=-\infty}^{\infty} \hat{z}_v \exp\left(\frac{2\pi i v p}{P}\right)$$ ..... Rumus 2.3

Koefisien $\hat{z}_v$ dikenal sebagai *Cartesian Fourier Descriptor* dari batas lengkung. Koefisien pertama

$$\hat{z}_0 = \frac{1}{P}\int_0^P z(p)dp = \frac{1}{P}\int_0^P x(p)dp + \frac{i}{P}\int_0^P y(p)dp$$ ..... Rumus 2.4

memberikan pusaran rata-rata (*mean vortex)* pada batas. Koefisien kedua mendeskripsikan lingkaran

$$z_1(p) = \hat{z}_1 exp\left(\frac{2\pi i p}{P}\right) = r_1 \exp(i\varphi_1 + 2\pi i p/P)$$ ..... Rumus 2.5

Radius r_1 dan titik awal pada *angle* φ_1 didapatkan dari $\hat{z}_1 = r_q exp(i\varphi_1)$. Koefisien $\hat{z}_{-1}$ juga dihasilkan pada lingkaran

$$z_{-1}(p) = r_{-1} exp(i\varphi_{-1} - 2\pi i p/P))$$ ..... Rumus 2.6

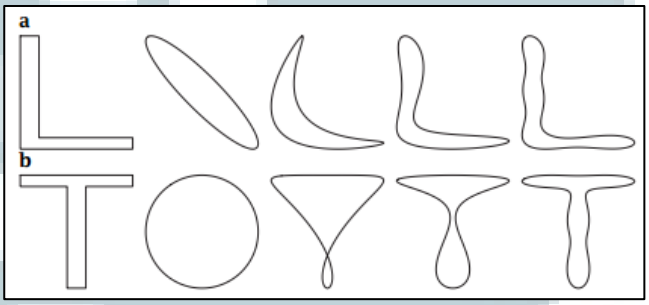
tetapi lingkaran ini ditelusuri dengan arah yang berlawanan (jarum jam). Dengan koefisien kompleks – dengan total empat parameter – elips dapat dibentuk dengan setengah sumbu bebas a dan b, orientasi ϑ dari sumbu

utama a, dan sumbu awal φ_0 pada elips. Sebagai contoh, diambil $\varphi_0 = \varphi_{-1} = 0$. Maka,

$$z_1 + z_{-1} = (r_1 + r_{-1}) \cdot cos\left(\frac{2\pi p}{P}\right) + i(r_1 - r_{-1}) sin\left(\frac{2\pi p}{P}\right) \quad \text{Rumus 2.7}$$

Fourier descriptor juga dapat dikomputasi dengan mudah dari *sampled bound-aries* z_n. Jika batas pinggir dari lengkung tertutup adalah P, sampel N harus diambil pada jarak yang sama dari *P/N* (gambar 2.4). Maka,

$$\hat{z}_v = \frac{1}{N} \sum_{n=0}^{N-1} z_n exp\left(-\frac{2\pi inv}{N}\right) \quad \text{Rumus 2.8}$$



Gambar 2.4 Rekonstruksi bentuk huruf L dan T dengan 2,3,4 dan 8 pasang *fourier descriptor*
Sumber: Jahne (2005, pp. 525)

2. *Polar Fourier Descriptor*

Merupakan salah satu pendekatan alternatif untuk *Fourier descriptor* dengan menggunakan *parameterization* lain pada garis batas. Alih-alih menggunakan panjang jalur p, sudut θ antara radius digambarkan dari titik tengah menuju titik pada garis batas dan garis x digunakan. Metode ini hanya membutuhkan *real-valued sequence, r*, dengan sampel *N* yang *equiangular* untuk mendeskripsikan batas. Koefisien dari *discrete fourier transform* pada urutan ini,

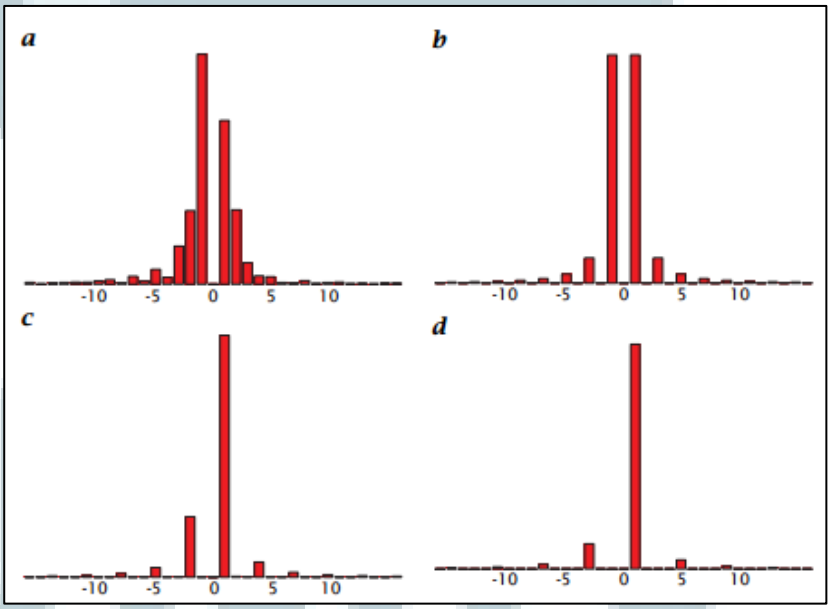
$$\hat{r}_v = \frac{1}{N} \sum_{n=0}^{N-1} r_n exp\left(-\frac{2\pi inv}{N}\right) \quad \text{Rumus 2.9}$$

dikenal sebagai *Polar Fourier descriptor* dari batas.

Koefisien pertama, $\hat{r}_0$, adalah sama dengan radius rata-rata. *Polar Fourier descriptor* tidak dapat digunakan untuk semua tipe batasan (*boundaries*).

3. *Symmetric Object*

Simetri dapat dengan mudah dideteksi dengan *Fourier descriptor*. Jika kontur memiliki simetri *m-rotational*, maka hanya $z_{1\pm vm}$ dapat tidak sama dengan nol. Demonstrasi pada gambar 2.6 merupakan *Fourier descriptor* dari garis lurus, segi tiga, dan bujur sangkar. Jika satu kontur merupakan cerminan kontur yang lain, maka *Fourier descriptor* adalah saling *conjugate complex.*



Gambar 2.5 Pengaruh simetri dari satu obyek terhadap *Fourier descriptor*-nya. **a** merupakan sebuah huruf L, **b** sebuah garis, **c** sebuah segi tiga, dan **d** sebuah bujur sangkar.
Sumber: Jahne (2005, pp. 526)

Dapat ditarik kesimpulan bahwa $|\hat{z}_{-v}| = |\hat{z}_v|$. Jika penelusuran dilakukan dari salah satu titik akhir, seimbang $\hat{z}_{-v} = \hat{z}_v$.

4. *Invarian Object Description*

Translation Invariance. Posisi dari sebuah obyek terbatas pada sebuah koefisien $\hat{z}_0$. Koefisien lainnya merupakan *translation invariance.*

Scale Invariance. Jika kontur diukur dengan koefisien α, semua *Fourier descriptor* juga diukur dengan α. Untuk objek dengan *non-zero area,* dan jika kontur ditelusuri berlawanan dengan arah jarum jam, koefisien pertama selalu tidak sama dengan nol. Pengukuran *Fourier descriptor* dapat dengan mudah dilakukan dengan $|\hat{z}_1|$untuk mendapatkan pengukuran *invariant shape descriptor*.

Rotation Invariance. Jika kontur diputar berlawanan dengan arah jarum jam dengan sudut φ_0, *Fourier descriptor* $\hat{z}_v$, dikalikan oleh *phase factor* $exp(iv\varphi_0)$ sesuai dengan *shift theorem* untuk *Fourier transform*. *Shift theorem* membuat konstruksi *rotation-invarian Fourier descriptor* menjadi lebih mudah. Model *invariance* ini memiliki isi dari dua jenis *Fourier descriptor* yang pertama.

Jika menormalisasikan *Fourier descriptor* yang lain dengan *phase* dan *magnitude* dari *Fourier descriptor* kedua, maka akan menghasilkan translasi, rotasi dan *scale invariant description* yang lengkap pada sebuah bentuk obyek. Perbedaan bentuk dapat diukur dengan menggunakan fakta bahwa *Fourier descriptor* membentuk *complex-valued vector.* Metrik untuk perbedaan bentuk didapatkan dari *magnitude* dari perbedaan vektor.

$$d_{zz'} = \sum_{v=-N/2}^{N/2-1} |\hat{z}_v - \hat{z}'_v|^2$$ ..... Rumus 2.10

Tergantung pada normalisasi apa yang digunakan pada *Fourier descriptor*, akan dilakukan *scale* dan/atau *rotation invariant* pada metrik ini.

Ismail (2010) mengatakan *Multiscale Fourier descriptor* dapat dibentuk dengan mengimplementasikan *discrete fourier transform* dari *Eq.*

$$a_n = \frac{1}{N}\sum_{i=0}^{N-1} u(t) \exp\left(-\frac{j2\pi nt}{N}\right) di\ mana\ u(t) = w$$ ..... Rumus 2.11

Proses ini akan memberikan hasil berupa kumpulan koefisien *fourier* $\{a_n\}$, yang merupakan representasi dari wilayah tanda tangan. Karena gambar dihasilkan melalui rotasi, translasi dan *scaling* (disebut dengan *similirarity transform* dari satu gambar) dari gambar yang sama, maka representasi gambar harus *invariant* terhadap operasi ini. Pemilihan titik awal pada batas gambar $u(t)$ harusnya tidak mempengaruhi representasinya.

Menurut Ismail (2010), berdasarkan teori *fourier*, koefisien *fourier* secara umum terdiri dari translasi, rotasi, *scaling*, dan perubahan titik awal sesuai dengan rumus sebagai berikut.

$$a_n = \exp(jnt) \times \exp(j\emptyset) \times x \times c \times a_n^{(0)} \quad n \neq 0$$ ..... Rumus 2.12

a_n^0 dan a_n secara berurutan adalah koefisien *fourier* dari gambar asli dan persamaan transformasi gambar. $exp(jnt)$, $exp(j\phi)$, dan s adalah istilah untuk perubahan titik awal, rotasi, dan skala. Kecuali komponen DC (a_0), semua koefisien lainnya tidak terpengaruh oleh translasi. Mempertimbangkan rumus

$$b_n = \frac{a_n}{a_0} = \frac{exp(jnt)\times exp(j\phi)\times c\times a_n^{(0)}}{exp(jt)\times exp(j\phi)\times c\times a_n^{(0)}} = \frac{a_n^{(0)}}{a_0^0} exp[j(n-1)t] =$$

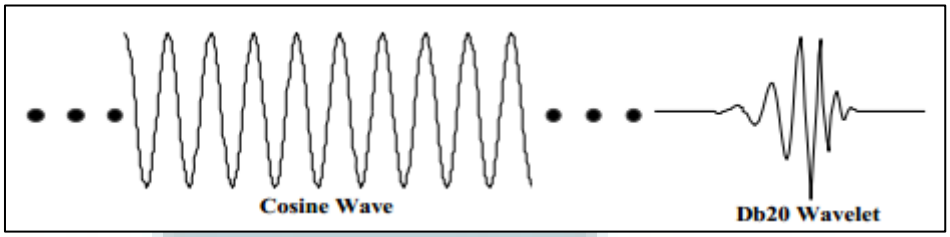
$$b_n^{(0)} exp[j(n-1)t]$$ ..... Rumus 2.13

$b_n$dan b_n^0 secara berurutan adalah koefisien *fourier* yang telah dinormalisasi dari gambar asal dan gambar asli. Koefisien yag telah dinormalisasi dari gambar asal b_n dan dari gambar asli b_n^0 hanya memiliki perbedaan pada $exp[j(n-1)t]$. Ismail (2010) mengatakan jika fase informasi diabaikan dan hanya menggunakan magnitude dari koefisien, maka $|b_n|$ dan $|b_n^0|$ adalah sama. Dengan kata lain, $|b_n|$ adalah invarian terhadap translasi, rotasi, skala, dan perubahan titik awal. Kumpulan magnitude dari koefisien fourier yang telah dinormalisasi dari gambar tanda tangan $\{|b_n|, 0 < n \leq N\}$ dapat digunakan sebagai deskripsi gambar tanda tangan, dinotasikan sebagai $\{FD_n\ 0 < n \leq N\}$.

2.4. Wavelet Transform

Menurut Ismail (2010) *Multi scale representatif* dari sebuah gambar tanda tangan dapat dicapai dengan menggunakan *wavelet transform.*

Wavelet, menurut Fugal (2009), merupakan bentuk gelombang dari durasi terbatas yang memiliki nilai rata-rata 0 (nol). Tidak seperti sinus yang secara teoritis membentang dari minus hingga positif tak berhingga. Sinus lebih lembut dan dapat diprediksi serta baik untuk mendeskripsikan sinyal frekuensi yang konstan. Sedangkan wavelet tidak merata, durasi terbatas, dan seringkali tidak simetris. Wavelet baik untuk mendeskripsikan anomali, getaran, dan kejadian lainnya yang dimulai dan diakhiri di dalam sinyal.

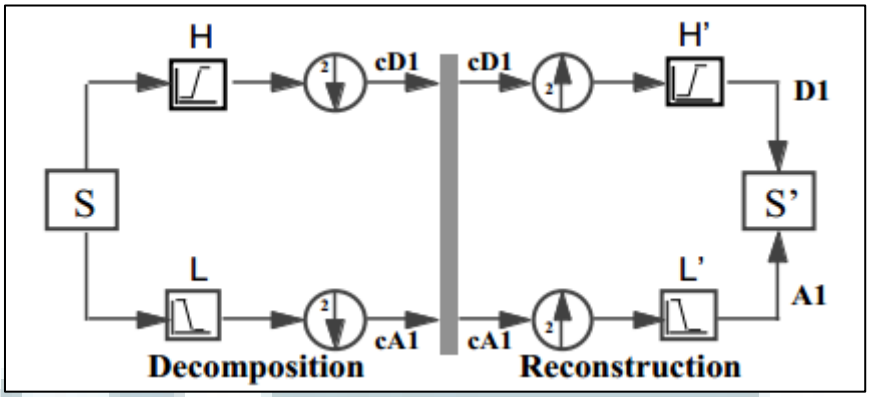


Gambar 2.6 Porsi dari panjang sinus tak hingga dan panjang wavelet terbatas
Sumber: Fugal (2009)

Wavelet Transform terbagi menjadi dua yaitu Continuous Wavelet Transform dan Discrete Wavelet Transform. Pada Continuous Wavelet Transform, suatu sinyal dengan energi terbatas diproyeksikan menjadi *band* frekuensi yang sifatnya kontinu. Pada kenyataannya, analisis setiap bagian dari koefisien *wavelet* yang sifatnya kontinu tidak mungkin untuk dikomputasi. Oleh karena itu, Discrete Wavelet Transform muncul untuk hanya menganalisis bagian-bagian koefisien *wavelet* tadi secara diskrit (Jayaraman, 2011, pp. 615).

Dalam bukunya Conceptual Wavelet, Fugal (2009) menjelaskan apabila dalam *Continuous Wavelet Transform (CWT)* wavelet dibentang secara terus menerus (dengan *integer steps*) dan *dyadically* (dengan faktor dari 2) pada *Undecimated Dicrete Wavelet Transform (UDWT)*. Sedangkan dalam *Discrete Wavelet Transform (DWT)* sinyal dikecilkan dan dibandingkan dengan wavelet filter yang tidak dapat diubah (*unchanged wavelet filter*). Hal ini dilakukan dengan *"downsampling by 2"*. *"Downsampling by 2"* memiliki arti menghilangkan setiap *sample* sinyal lainnya.

Sebagai contoh, jika dimiliki deretan angka (sinyal) [5 4 3 2] diubah menjadi [5 3] (atau [4 2] tergantung dari mana perhitungan dimulai). Proses ini juga dapat dirujuk ke terminologi wavelet sebagai *"decimation by 2"*.



Gambar 2.7 *Single level conventional DWT*

Sumber: Fugal (2009)

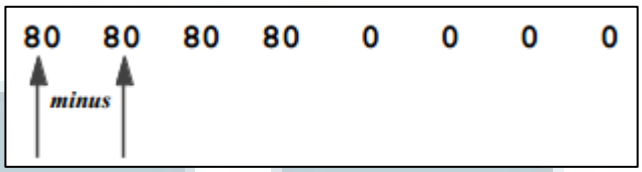
Fugal (2009) menjelaskan bahwa kualitas luar biasa yang dimiliki DWT adalah, dengan menggunakan wavelet filter yang tepat (H, H', L, dan L') maka rekonstruksi dapat dilakukan bahkan dengan *aliasing. Aliasing* adalah terdapat 2 atau lebih sinyal yang memiliki nilai sampel yang sama.

Terdapat dua hal yang harus diperhatikan pada saat menggunakan DWT dibanding dengan UDWT yang lebih sederhana yakni:

1. *Downsampling by 2* dalam DWT dapat menghasilakan *aliasing* (membuang setengah dari sampel dapat menghasilkan sinyal yang salah).
2. Perubahan bentuk (*transform*) ini tidak *shift-invariant* (terkadang disebut dengan *time invariant*)

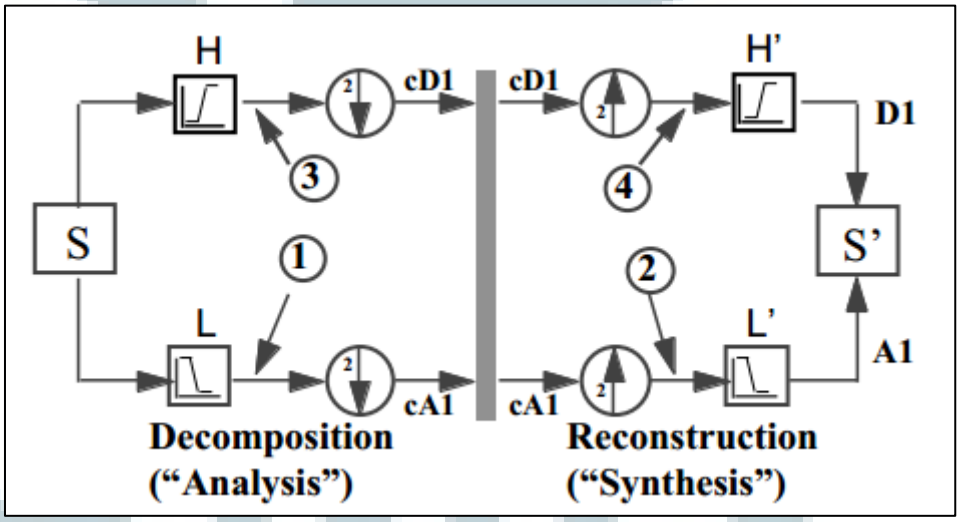
Dalam bukunya yang berjudul *Conceptual Wavelet,* Fugal (2009) menjelaskan skenario penggunaan wavelet dan filternya dengan mengambil contoh menggunakan nilai ujian. Nilai ujian tersebut dapat diibaratkan sebagai gelombang dan terdapat sebuah parameter yang digunakan sebagai bank filter. Bank filter yang digunakan adalah Haar wavelet filter sederhana [1,-1]. Gambar

2.8 menjelaskan contoh proses yang dilakukan dalam mengimplementasikan *wavelet filter*.



Gambar 2.8 Perbandingan Haar Wavelet filter [1, -1] dengan 2 nilai ujian pertama.
Sumber: Fugal (2009)

Seperti gambar 2.8 proses perbandingan terus dilanjutkan (dengan melakukan pergesaran deret satu per satu) hingga deret angka telah habis dan akan menghasilkan [0, 0, 0, 80, 0, 0, 0]. Hasil ini, menurut Fugal (2009), sudah hampir sama dengan apa yang dihasilkan oleh MATLAB hanya bedanya MATLAB memiliki 2 pembanding tambahan di deret angka paling depan dan belakang dengan nilai 0. Sehingga akan menghasilkan deret [-80, 0, 0, 0, 80, 0, 0, 0, 0].



Gambar 2.9 Diagram sinyal *single level DWT* konvesional.
Sumber: Fugal (2009)

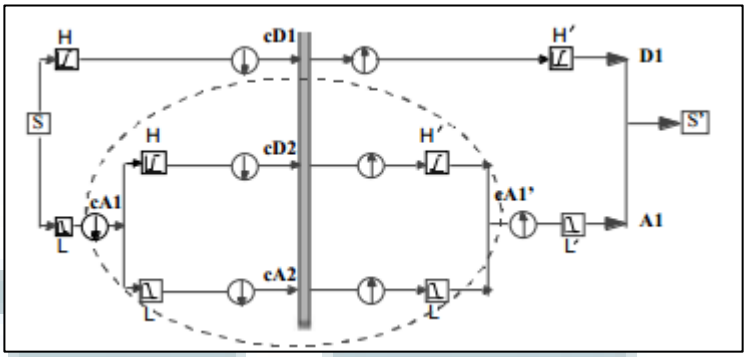
Jika diperhatikan pada gambar 2.9, maka dengan *downsampling*, detil koefisien (cD1) dan pendekatan koefisien (cA1) adalah setengah dari panjang sinyal awal, S.

Pada percobaan dengan DWT, Fugal (2009) mengkonvolusikan sinyal dengan L = [1,1]. L adalah bank filter. Dengan data yang sama yakni S = [80, 80, 80, 80, 0, 0, 0, 0] dihasilkan nilai conv(S, [1, 1]) = 80 160 160 160 80 0 0 0 0. Hasil tersebut sudah sama apabila melakukan pemrosesan dengan UDWT. Karena DWT, maka pada hasil tersebut akan dilakukan *downsampling* dengan 2 sehingga menghasilkan cA1=dyaddown(conv(S, [1, 1]) = 160 160 0 0. Fugal (2009) mengambil contoh dengan MATLAB menggunakan *number generator* yang kemudian menghasilkan sinyal dan diproses dengan menggunakan DWT menghasilkan nilai pada gambar 2.10.

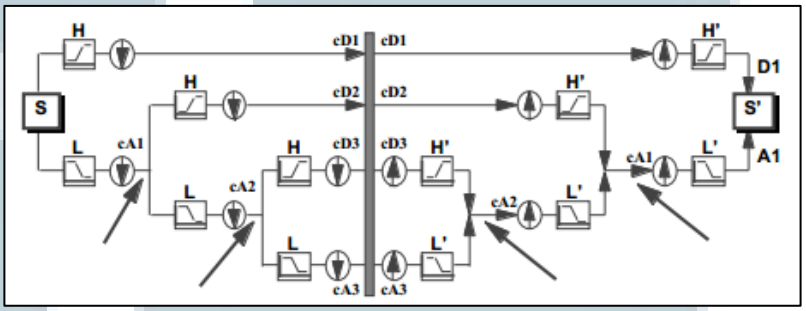
```
cD1 = dyaddown(conv(S,[-1 1])) = 1.2451   -0.4521
-0.6523    4.8274

D1 = conv(dyadup(cD1),[1 -1]) =  0  1.2451  -1.2451  -
0.4521  0.4521  -0.6523  0.6523  4.8274  -4.8274  0

cA1 = dyaddown(conv(S,[1  1])) = 0.7138   -0.6446
-2.1090   -1.0554

A1 = conv(dyadup(cA1),[1 1]) = 0  0.7138  0.7138
-0.6446 -0.6446 -2.1090 -2.1090 -1.0554  -1.0554  0

SUM = D1 + A1 =  0    1.9589   -0.5312   -1.0967   -
0.1925   -2.7613   -1.4567 3.7720   -5.8828   0
```

Gambar 2.10 Hasil perhitungan *single level DWT*
Sumber: Fugal (2009)

Dalam pemrosesan *multi-level DWT*, Fugal (2009) menggambarkan 2-level DWT dan 3-level DWT sebagai berikut.



Gambar 2.11 2-level DWT
Sumber: Fugal (2009)

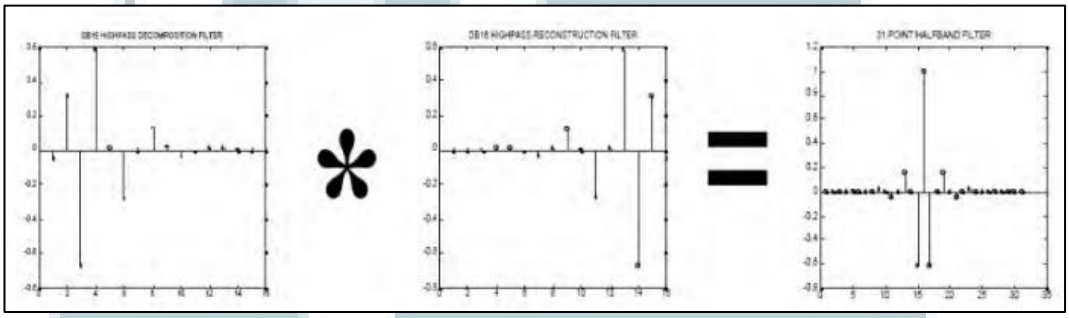


Gambar 2.12 3-level DWT
Sumber: Fugal (2009)

Pada gambar 2.12, dua panah paling bawah memperlihatkan bahwa cA2 direkonstruksi dengan menggunakan *embedded single-level DWT*. Tanda panah di atasnya memperlihatkan bahwa cA1 direkonstruksi dengan menggunakan *embedded single-level DWT* (dengan catatan cA2 telah selesai direkonstuksi). Setelah selesai melakukan rekonstruksi terhadap cA2, maka dapat melakukan rekonsturksi S dengan sempurna.

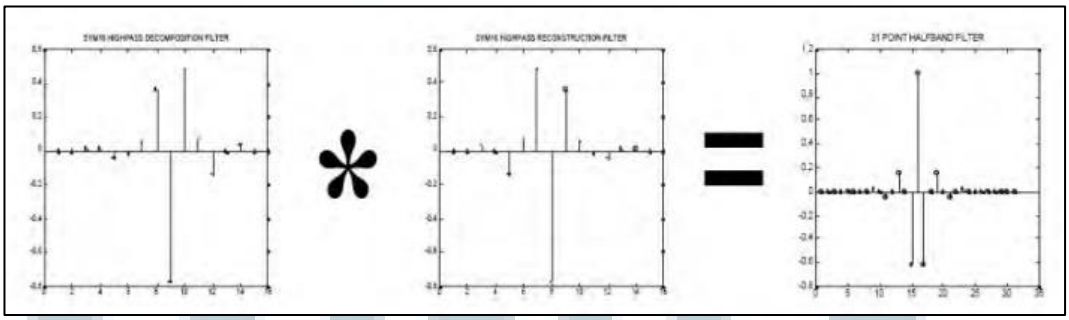
Fugal (2009) mengatakan, berdasarkan namanya, symlet wavelet transform atau symlet lebih simetris dibandingkan dengan daubechies wavelet. Fugal (2009) menjelaskan bahwa *halfband* filter dapat di "faktorkan" dengan lebih dari satu cara untuk mendekomposisi filter dan merekonstruksi filter yang,

ketika *convolved*, dapat memproduksi *halfband* filter yang sama. Untuk *halfband* filter yang lebih panjang (23 poin atau lebih), dapat diproduksi filter alternatif selain daubechies yang lebih simetris. Fugal (2009) mencontohkan ketika terdapat 31 poin *highpass halfband* filter dapat difaktorkan menjadi 2 buah 16 poin filter Daubechies 16 H dan H' seperti pada gambar 2.13.



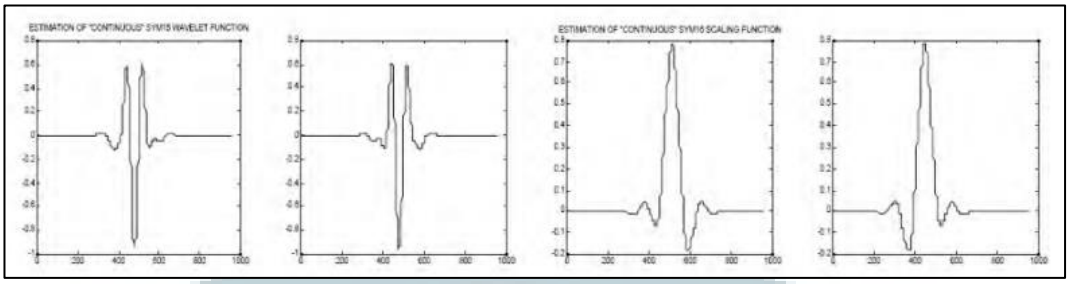
Gambar 2.13 Dua buah 16 poin Db16 menjadi 31 poin *highpass* filter
Sumber : Fugal (2009)

Fugal (2009) menjelaskan bahwa 31 poin highpass filter tersebut juga dapat dibentuk menjadi dua buah 16 poin filter yang lebih simetris (walaupun tidak sempurna) seperti gambar 2.14.



Gambar 2.14 Dua buah 16 poin Sym16 menjadi 31 poin highpass filter
Sumber : Fugal (2009)

Representasi sinyal Symlet wavelet transform digambarkan oleh Fugal (2009) seperti pada gambar 2.15.



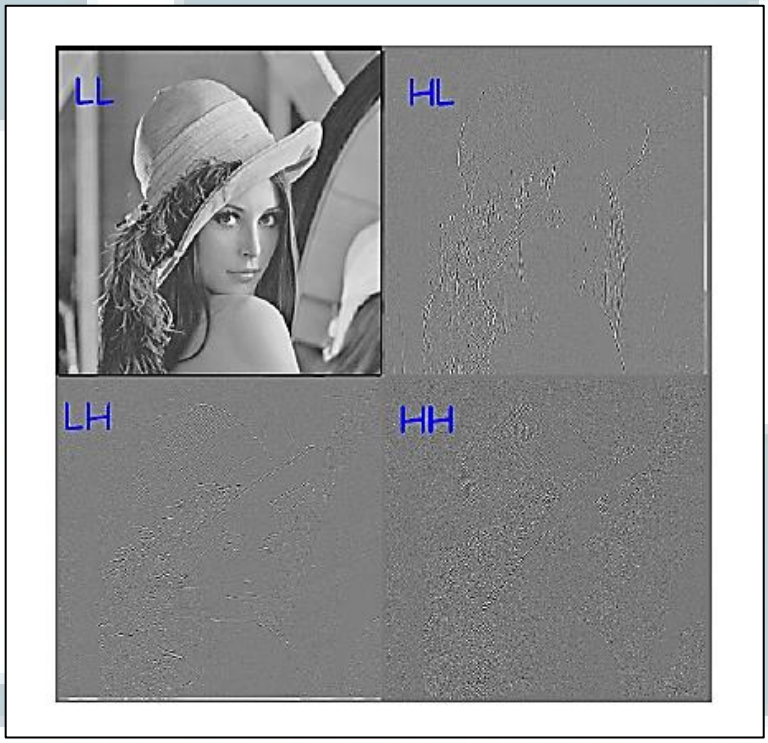
Gambar 2.15 Bentuk gelombang sinyal symlet
Sumber : Fugal (2009)

Fugal (2009) menjelaskan bahwa pada grafik paling kiri pada gambar 2.15 didapatkan dengan melakukan *upsampling* secara berulang terhadap highpass filter rekonstruksi 16 poin symlet, *hirsym*, dan *lowpass filtering* pada setiap langkah dengan *lorsym.* Sedangkan pada grafik kedua pada gambar 2.15 dibentuk dari proses interpolasi yang sama dimulai dengan *hidsym* dan *lowpass filtering* dengan *lodsym*. Fugal (2009) mengatakan bahwa grafik pertama dan kedua mirip dengan H' dan H pada daubechies wavelet secara terbalik. Grafik ketiga pada gambar 2.15 merupakan estimasi dari fungsi skala "*continuous"* yang didapat dengan interpolasi *lorsym* (*lorsym* juga digunakan sebagai *lowpass filter*). Grafik terakhir pada gambar 2.15 mirip dengan L' dan L untuk daubechies wavelet secara terbalik dan didapatkan dengan melakukan *dyadic upsampling* dari *lodsym* dengan melakukan *filtering* dengan *lodsym* pada setiap langkahnya.

Menurut Fugal (2009) dengan mendekati simetris, semakin besar symlet (Sym12, Sym16, dan seterusnya) juga memiliki fase yang mendekati linear. Selain simetri dan *phase,* symlet berbagi *properties* yang sama dengan daubechies wavelet. Kedua wavelet ini menjadi lebih reguler dengan nilai N yang semakin besar ("SymN"), kedua wavelet ini juga memiliki *compact support* seperti Daubechies dengan nilai N, wavelet ini juga memiliki jumlah *vanishing moment*

yang sama dengan keluarga DaubechiesN, dan memiliki rekonstruksi sempurna serta kemampuan *alian cancellation* yang memungkinkan jenis wavelet ini digunakan baik pada *Continuous Wavelet Transform* (CWT) maupun *Discrete Wavelet Transfotm* (DWT).

Liu (2010) mengatakan bahwa hasil transformasi Discrete Wavelet Transform untuk gambar digital dibagi menjadi 4 *band* yaitu LL, LH, HL, dan HH. L merupakan singkatan dari *low*, sedangkan H merupakan singkatan dari *high*. Setiap band merepresentasikan tingkat frekuensi dari hasil transformasi. Gambar 2.16 menunjukkan hasil transformasi Discrete Wavelet Transform sebanyak satu kali iterasi.



Gambar 2.16 Dekomposisi gambar hasil Discrete Wavelet Transform
Sumber: Liu (2010, pp. 23).

Getreuer (2006) menjelaskan bahwa sudah lebih dari dua dekade wavelet mendapatkan banyak popularitas untuk digunakan sebagai alat bantu untuk berbagai disiplin ilmu. Getreuer (2006) menjelaskan pula bahwa dapat menjadi sulit untuk mendapatkan koefisien filter bahkan pada jenis wavelet yang paling sering digunakan. Menurut Getreuer (2006) Symlet merupakan Daubechies wavelet dalam bentuk yang kurang lebih simetris, wavelet ortogonal dimana fungsi skala adalah dekat dengan simetris. Getruer (2006) menjelaskan bahwa untuk mendapatkan koefisien wavelet Symlet adalah dengan rumus berikut.

$h(z) = \sum_k h_k z^{-k}$ ..... Rumus 2.14

$g(z) = zh(-z^{-1})$ ..... Rumus 2.15

$\tilde{h}(z) = h(z^{-1})$ ..... Rumus 2.16

$\tilde{g}(z) = g(z^{-1})$ ..... Rumus 2.17

Menurut Ismail (2010), dalam *Discrete Wavelet Transform (DWT),* koefisien *wavelet* dari sebuah image f(x,y) di definisikan sebagai

$W_\vartheta(j_0, m, n) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \vartheta_{j_0,m,n}(x,y)$ ..... Rumus 2.18

$W_\Psi^i(j_0, m, n) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \psi_{j_0,m,n}^i(x,y) \qquad h = \{H, V, D\}$

Rumus 2.19

Di mana J_0 adalah skala awal.

Permasalahan dengan koefisien yang didapat dari *wavelet transform* adalah fakta bahwa mereka bukanlah *rotation invariant.* Dimensi dari satu vektor juga tergantung seberapa besar gambar tanda tangan yang telah diambil. Maka dari itu, koefisien vektor dari tanda tangan yang berbeda tidak dapat secara langsung dibandingkan pada saat pengambilan gambar. Solusi ini bertujuan untuk

mengimplementasikan *fourier transform* pada koefisien yang didapat dari *wavelet transform.* Dengan cara ini, representasi *Multi scale signature* dapat ditransformasikan menjadi domain frekuensi, di mana operasi normalisasi dan pencocokan mudah dilakukan.

2.5. Distance Measures

Dalam penelitiannya, Ismail (2010) melakukan proses pengukuran jarak atau *distance measures.* Dengan menjadikan X dan Y menjadi vektor pada panjang n, maka dapat dihitung jarak antara kedua vektor ini.

1. Minkowski Distance

$$d(X,Y) = L_p(X,y) = \left(\sum_{i=1}^{n}|x_i - y_i|\right)^{i/p} \quad \text{Rumus 2.20}$$

2. Manhattan Distance (L_1 *matrices, City block distance*)

$$d(X,Y) = L_{p=1}(X,y) = \sum_{i=1}^{n}|x_i - y_i| \quad \text{Rumus 2.21}$$

3. Euclidean Distance (L_2 *matrices*)

$$d(X,Y) = L_{p=2}(X,y) = \sqrt{\sum_{i=1}^{2}(X_i - y_i)^2} \quad \text{Rumus 2.22}$$

4. Angle-based Distence

$$d(X,Y) = -cos(X,Y) \quad \text{Rumus 2.23}$$

$$-cos(X,Y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2}} \quad \text{Rumus 2.24}$$

5. Correlation coefficient-based Distance

$$d(X,Y) = -r(X,Y) \quad \text{Rumus 2.25}$$

$$r(X,Y) = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \ \sum_{i=1}^{n} y_i}{\sqrt{\left(n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right)\left(n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2\right)}} \quad \text{Rumus 2.26}$$

6. Modified Manhattan Distance

$$d(X,Y) = \frac{\sum_{i=1}^{n}|x_i - y_i|}{\sum_{i=1}^{n}|x_1| \sum_{i=1}^{n}|y_1|} \quad \text{Rumus 2.27}$$

7. Modified SSE-based Distance

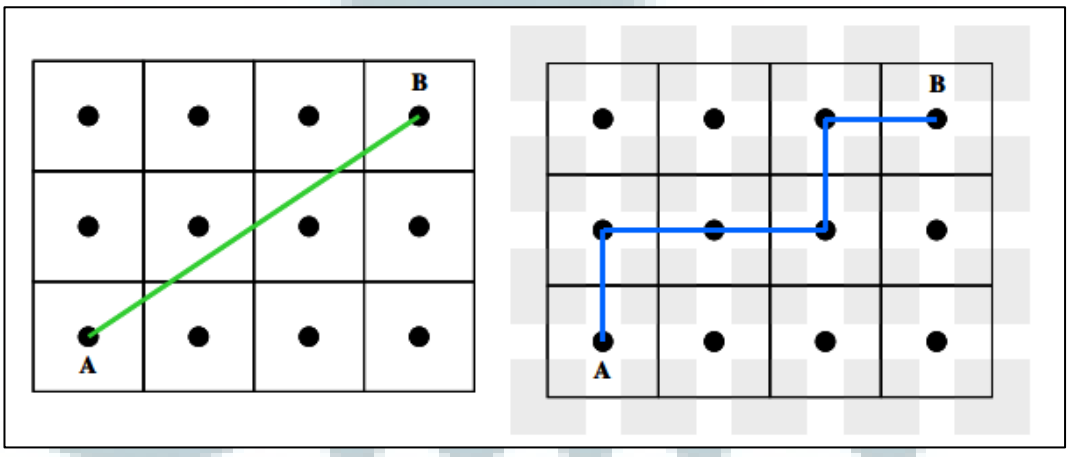
$$d(X,Y) = \frac{\sum_{i=1}^{n}(x_i - y_i)^2}{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2} \quad \text{Rumus 2.28}$$

Menurut Caballero (2006) Manhattan Distance atau yang biasa dikenal dengan Taxicab Geometry, ditemukan oleh Hermann Minkowski pada abad ke-19, adalah merupakan bentuk geometri dimana fungsi jarak biasa atau *metric of euclidean distance* diubah dengan *metric* baru dimana jarak antara dua titik adalah jumlah dari perbedaan koordinat kedua titik tersebut diabsolutkan (Krause, 1987). Secara formal, Manhattan Distance, atau dapat juga disebut sebagai L1-distance dapat didefinisikan sebagai jarak antara dua titik pada ruang Eucledian dengan sistem koordinat Cartesian tetap didefinisikan sebagai jumlah dari panjang proyeksi dari segment garis antara titik menjadi koordinat sumbu Sebagai contoh, Manhattan distance T antara titik A dengan koordinat (X_a, Y_a) dan titik B pada (X_b, Y_b) adalah

$$T = |X_b - X_a| + |Y_b - Y_a| \quad \text{Rumus 2.29}$$

Sedangkan Eucledian Distance E untuk titik yang sama dikalkulasikan dengan

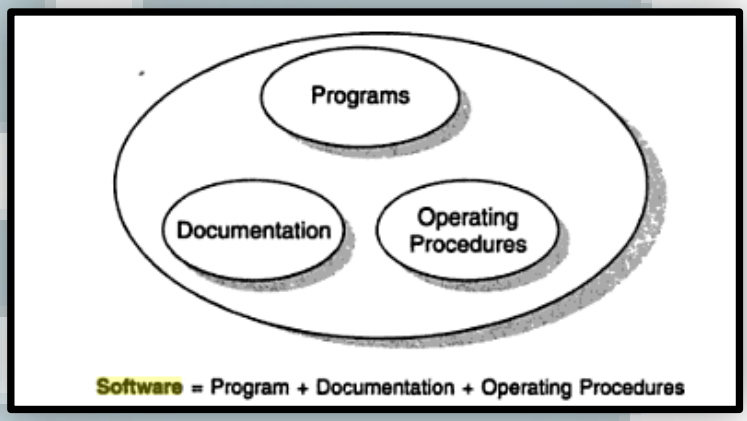
$$E^2 = (X_b - X_a)^2 + (Y_b - Y_a)^2 \quad \text{Rumus 2.30}$$



Gambar 2.17 Eucledian Distance (Kiri) dan Manhattan Distance (Kanan)
Sumber : Caballero (2006)

2.6. Peranti Lunak

Menurut Aggarwal (2006, pp.5) peranti lunak merupakan lebih dari sebuah program. Peranti lunak terdiri dari program, dokumentasi dari segala aspek program, dan prosedur untuk melakukan instalasi serta mengoperasikan sistem peranti lunak. Komponen sebuah peranti lunak terdapat dalam gambar 2.18.



Gambar 2.18 Komponen peranti lunak
Sumber: Aggarwal (2006, pp. 6).

Program adalah bagian dari peranti lunak dan itu akan menjadi peranti lunak jika dokumentasi dan petunjuk prosedur pengoperasian juga disiapkan. Program merupakan kombinasi dari *source code* dan *object code.*

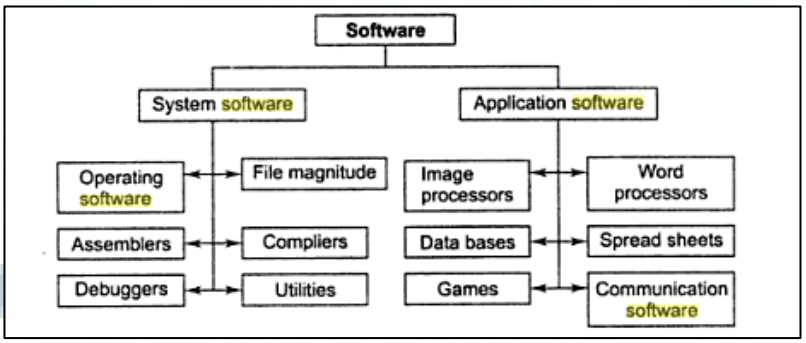
Sedangkan menurut Agarwal (2007, pp. 1) peranti lunak adalah set instruksi untuk menerima sebuah *input* dan memanipulasinya untuk menghasilkan *output* yang diinginkan dengan kondisi fungsi dan performa ditentukan oleh pengguna peranti lunak. Peranti lunak juga termasuk di dalamnya dokumen, seperti buku petunjuk penggunaan, bertujuan agar pengguna mengerti sistem dari peranti lunak. Peranti lunak sekarang ini terdiri dari *source code, executeable,*

design document, operation and system manual, dan *installation and implementation manual.*

Masih menurut Agarwal (2007, pp. 2), dijelaskan pentingnya sebuah peranti lunak. Menurutnya, peranti lunak sudah menjadi alat penggerak. Peranti lunak menggerakkan bisnis dengan memberikan keputusan, melayani sebagai basis untuk investigasi sains modern dan pemecahan masalah teknis. Peranti lunak juga dapat ditanamkan dalam berbagai macam sistem seperti transportasi, medis, telekomunikasi, militer, industri, hiburan, produk kantor, dan lain-lain.

Dalam bukunya, Agarwal (2007, pp. 2) membagi tipe software menjadi dua kategori:

1. Peranti lunak sistem. Peranti lunak ini termasuk sistem operasi dan segala keperluan yang menjalankan fungsi dari sebuah komputer.
2. Peranti lunak aplikasi. Peranti lunak ini terdiri dari program yang bekerja secara nyata untuk pengguna. Sebagai contoh, pemrosesan kata, *spread sheet*, dan sistem manajemen basis data termasuk dalam kategori peranti lunak aplikasi.



Gambar 2.19 Tipe peranti lunak
Sumber: Agarwal (2007, pp. 2).

Agarwal (2007, pp. 5) juga mengklasifikasikan peranti lunak ke dalam dua kelas yakni:

1. Peranti lunak umum. Peranti lunak umum didisain untuk pelanggan pasar yang luas di mana kebutuhannya sangat umum, hampir stabil dan sangat dimengerti oleh *software engineer*.
2. Perangkat lunak disesuaikan. Produk yang disesuaikan dikembangkan untuk pelanggan yang wilayah, lingkungan, dan kebutuhan menjadi unik dan tidak puas dengan perani lunak umum.

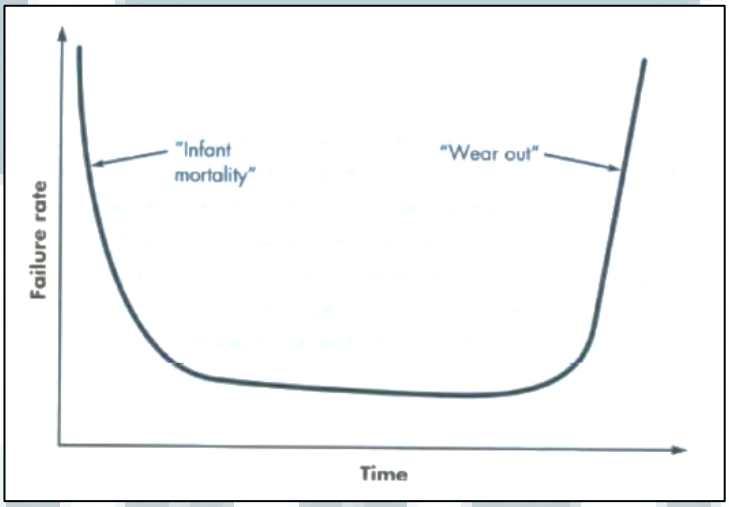
Kevin Arian (2013), dengan mengutip dari Pressman (2009, pp. 37-39) menyimpulkan beberapa karakteristik piranti lunak yang membedakannya dari piranti keras yaitu sebagai berikut.

1. Piranti lunak dikembangkan, tidak dimanufaktur.

   Walaupun ada beberapa kesamaan di antara pengembangan piranti lunak dan manufaktur piranti keras, secara fundamental kedua aktivitas tersebut sangat berbeda. Pada kedua aktivitas, kualitas yang baik tentu dicapai lewat desain yang baik. Akan tetapi, pada proses manufaktur piranti keras, ada beberapa masalah kualitas yang sering muncul dan tidak mudah untuk diperbaiki. Lain halnya dalam pengembangan piranti lunak, masalah-masalah yang muncul relatif lebih mudah untuk diperbaiki. Kedua aktivitas ini juga tergantung pada sumber daya manusia dan memerlukan konstruksi produk yang baik. Akan tetapi, pendekatan untuk keduanya jelas berbeda karena proyek piranti lunak tidak bisa dikerjakan selayaknya proyek manufaktur.

2. Piranti lunak tidak dapat habis walaupun dipakai terus-menerus.

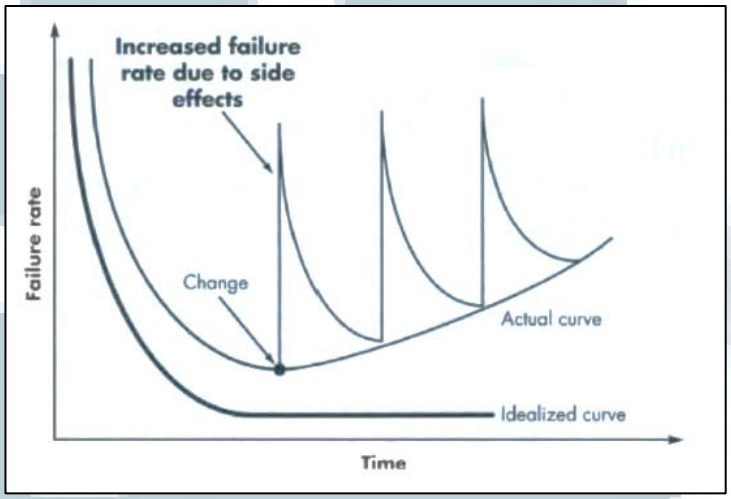
Dalam dunia piranti keras, pengalaman mengindikasikan bahwa perangkat ini akan melewati fase rata-rata kesalahan yang tinggi di awal pengembangannya. Kesalahan-kesalahan ini namun dapat diperbaiki dan piranti keras dapat hidup pada fase aman. Akan tetapi, seiring berjalannya waktu, rata-rata kesalahan tersebut akan meningkat lagi karena efek kumulatif dari debu, getaran, penyalahgunaan, temperatur ekstrim, atau efek lingkungan lainnya. Secara singkat, piranti keras dapat habis seiring waktu pemakaiannya. Gambar 2.1 menunjukkan hubungan rata-rata kesalahan sebuah piranti keras terhadap waktu.



Gambar 2.20 Hubungan rata-rata kesalahan piranti keras terhadap waktu
Sumber: Pressman (2009, pp. 37).

Lain halnya dengan piranti lunak, efek-efek lingkungan sama sekali tidak mempengaruhi kinerjanya. Oleh karena itu, grafik rata-rata kesalahan untuk piranti lunak relatif lebih ideal. Kesalahan-kesalahan pada software biasanya terdeteksi pada fase awal. Perbaikan-perbaikan yang dilakukan membuat daya hidup suatu piranti lunak jauh lebih baik. Akan tetapi, piranti

lunak tidak selamanya memenuhi kebutuhan pengguna. Selama masa hidupnya tentu akan ada perubahan-perubahan yang membuat grafik rata-rata kesalahannya meningkat sedikit demi sedikit. Gambar 2.2 menunjukkan hubungan rata-rata kesalahan sebuah piranti lunak terhadap waktu.



Gambar 2.21 Hubungan rata-rata kesalahan piranti lunak terhadap waktu
Sumber: Pressman (2009, pp. 38).

Dapat disimpulkan bahwa piranti lunak tidak habis dipakai secara terus-menerus akan tetapi kinerjanya bisa menurun akibat perubahan-perubahan. Kerusakan dalam piranti lunak hanya dapat diperbaiki dengan pemrograman. Lain halnya dengan piranti keras, apabila rusak komponennya dapat diganti dengan *spare-part* lain.

3. Industri bergerak ke arah standarisasi konstruksi komponen sedangkan piranti lunak dapat dikustomisasi pengembangannya.

Seiring dengan berevolusinya ilmu *engineering*, sekumpulan standar desain pun banyak diciptakan. Para ilmuwan bekerja dengan ide dasar untuk menciptakan komponen-komponen standar yang dapat digunakan secara umum untuk berbagai piranti keras. Hal ini juga

menginspirasi para ilmuwan piranti lunak, akan tetapi, yang membedakannya adalah standar-standar penciptaan komponen yang dapat digunakan terus-menerus itu dapat dikustomisasi (*override*) sesuai kebutuhan.

Kevin Arian (2013) dengan mengutip Sommerville (2007, pp. 7) mengatkan bahwa rekayasa piranti lunak (*software engineering*) adalah disiplin ilmu insinyur yang fokus pada semua aspek dari produksi piranti lunak dari tahap awal yaitu spesifikasi sistem sampai pada tahap pemeliharaan saat sistem sudah digunakan. Diungkapkan pula bahwa perlu diperhatikan bahwa *computer science* berbeda dengan *software engineering*. *Computer science* berfokus pada teori dan metode yang mendasari komputer dan sistem piranti lunak, sedangkan *software engineering* berfokus pada masalah praktis dalam produksi piranti lunak. Oleh karena itu ada dua kata kunci yang perlu diperhatikan dalam terminologi rekayasa piranti lunak.

Pertama adalah disiplin ilmu insinyur yang membuat sesuatu bekerja dengan mengaplikasikan teori, metode, dan berbagai peralatan yang ada. Walaupun demikian, seorang insinyur tidak berhenti pada teori atau metode yang ada. Insinyur terus mencari solusi untuk suatu permasalahan yang belum ada teori untuk memecahkannya. Kedua adalah aspek-aspek dari produksi perangkat lunak.

Menurut Sundar (2010, pp. 9-10), ketika suatu spesifikasi piranti lunak sudah didapatkan, pengembangan piranti lunak akan melalui beberapa fase yang secara umum adalah tahap spesifikasi, desain, dan implementasi. Tim pengembang harus mendefinisikan suatu model siklus hidup dari pengembangan.

Setiap anggota pengembang harus memiliki pemahaman yang jelas dari siklus hidup yang diadopsi agar tidak terjadi kesalahan dan kekacauan dalam pengembangan piranti lunak.

2.8 Android

Seperti yang tercatat pada halaman web android.com, Android adalah platform perangkat bergerak paling populer di dunia. Banyak hal yang dapat dilakukan dengan android. Menggunakan fasilitas dari aplikasi google, buku, musik, dan lain-lain. Perangkat android sudah cerdas dan menjadi lebih cerdas dengan fitur yang tidak dapat ditemukan di platform lain.

Pada tahun 2005, Elgin (2005) membuat berita mengenai pembelian android yang dilakukan oleh google. Di dalam laporan tersebut pula diceritakan bahwa Android adalah sistem operasi yang berbasiskan Linux dan didesain untuk perangkat bergerak yang memiliki fasilitas layar sentuh seperti ponsel cerdas dan perangkat komputer tablet.

Rubin (2007) dalam *blog*-nya mendeskripsikan android sebagai platform pertama yang benar-benar terbuka serta komprehensif untuk perangkat bergerak. Ini meliputi sistem operasi, antarmuka pengguna, dan aplikasi. Semua perangkat lunak ini berjalan pada telepon genggam tetapi tanpa hambatan hak milik yang menentang inovasi perangkat bergerak.

Menurut Meier (2012) sekarang ini android adalah ekosistem yang terdiri dari kombinasi 3 komponen:

1. Gratis, sistem operasi dengan sumber terbuka (*Open-source)* untuk perangkat *embedded.*

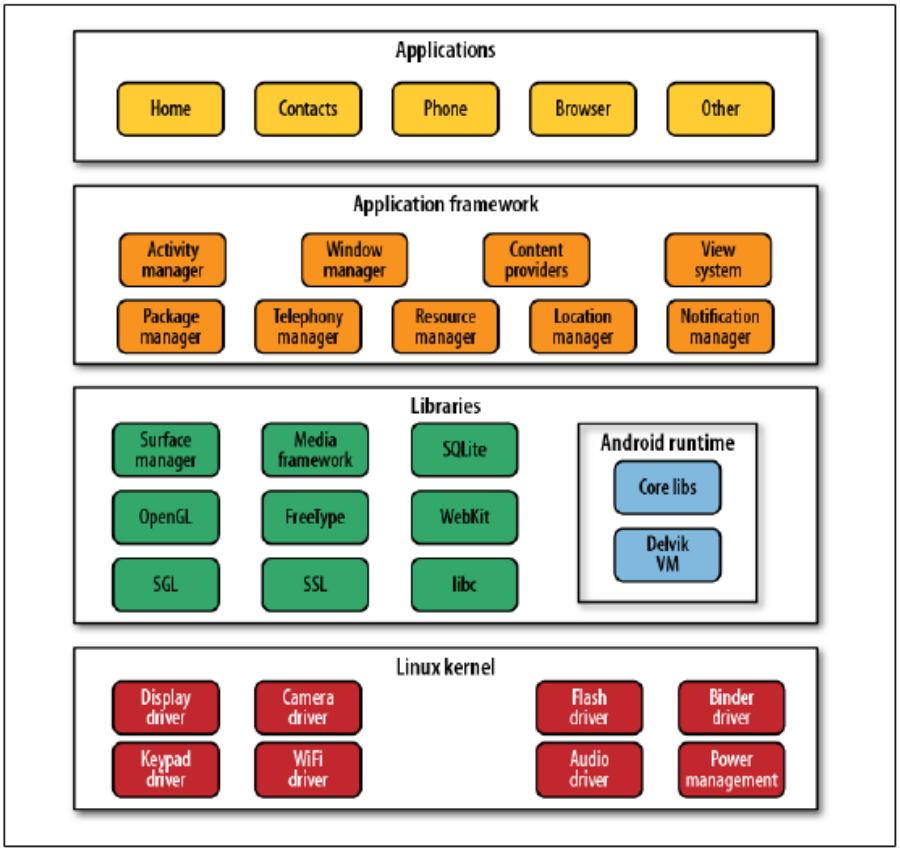
2. Platform pengembangan yang o*pen-source* untuk membuat aplikasi.
3. Perangkat, secara detail telepon genggam, yang menjalankan sistem operasi android dan aplikasi yang dibuat untuknya.

Secara spesifik, android terdiri dari beberapa bagian yang penting dan saling bergantung, seperti:

1. *Compability Definition Document (CDD)* dan *Compability Test Suite (CTS)* yang mendeskripsikan kapabilitas yang dibutuhkan sebuah perangkat untuk mendukung *stack* perangkat lunak.
2. *Kernel* sistem operasi Linux yang menyediakan *low-level interface* dengan perangkat keras, pengaturan memori, dan kontrol proses, semua dioptimalkan untuk perangkat bergerak dan perangkat *embedded.*
3. *Library* sumber terbuka untuk pengembangan aplikasi termasuk SQLite, WebKit, OpenGL, dan *media manager.*
4. *Run Time* untuk menjalankan dan menampung aplikasi android, termasuk *Dalvik Virtual Machine (DVM)* dan *core libraries* yang menyediakan fungsionalitas spesifik android. *Rum Time* didesain untuk menjadi kecil dan efisien untuk digunakan di perangkat bergerak.
5. *Framework* aplikasi yang secara *agnostically* membuka servis dari sistem kepada *layer* aplikasi, termasuk *window manager, location manager,* basis data, telepon dan sensor.
6. Kumpulan dari aplikasi utama *pre-installed*

7. Sebuah *Software Development Kit (SDK)* yang digunakan untuk membuat aplikasi, termasuk alat-alat (*tools*) yang berkaitan, *plug-ins*, dan dokumentasi.

Gargenta (2011) mengatakan Android terdiri dari kernel yang berdasar pada Linux kernel 2.6 dan Linux kernel 3.x (untuk Anroid versi 4 ke atas). Android dilengkapi dengan *middleware*, *libraries*, dan API yang ditulis menggunakan bahasa C dan perangkat lunak aplikasi yang berjalan pada sebuah *framework* aplikasi, termasuk *library* yang cocok dengan Java. Android akan mengeksekusi program di atas Dalvik virtual Machines yang akan melakukan kompilasi terhadap Java *byte code.*



Gambar 2.22 Android *stack*
Sumber: Gargenta (2011, pp. 8).