BAB III

LANDASAN TEORI

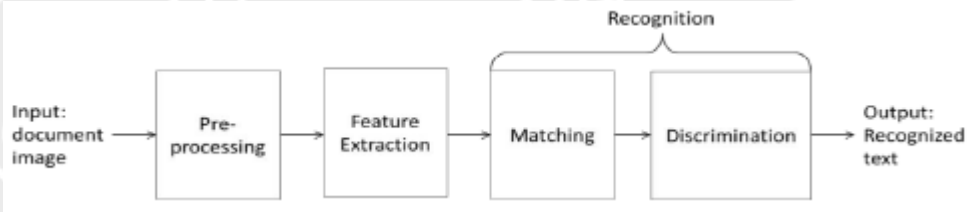
Pada bab ini akan dijelaskan mengenai teori dasar yang digunakan oleh penulis sebagai acuan dalam pengembangan aplikasi.

3.1 Optical Character Recognition

Optical Character Recognition(OCR) dapat didefinisikan sebagai analisis elektronik dari sebuah citra dalam upaya mengidentifikasikan wilayah yang memiliki informasi tekstual dan ekstraksi atau pengenalan teks dari citra yang diberikan(Jain et al, 2013). Definisi OCR secara sederhana adalah proses pengenalan teks hasil *print*, tulisan tangan, hasil mesin ketik dan sebagainya dari suatu berkas *hard copy* yang diubah ke berkas digital(Lian, 2009). Ciri aplikasi OCR yang telah sering digunakan berkaitan dengan pendokumentasian citra. Citra dari halaman-halaman buku, dokumen sah, catatan medis, dan sebagainya diperoleh melalui *scanner flatbed* kemudian diproses oleh OCR(Lian, 2009). Pengenalan optis dilakukan secara *off-line* setelah penulisan atau proses pencetakan selesai, berkebalikan dengan pengenalan on-line dimana komputer mengenali karakter saat digambarkan. Hasil pengenalan karakter dari tulisan tangan maupun cetak komputer tergantung pada kualitas dari input data tersebut(Eikvil,1993). Terdapat beberapa variabel yang mempengaruhi kualitas data input seperti yang sudah disebutkan, diantaranya kejelasan bentuk karakter(tulisan tangan) dan atau *font*, kualitas

alat cetak(*printer*), kualitas citra(piksel), latar, dan warna. Semakin baik mutu data input, maka performa OCR akan mendekati kemampuan manusia dalam mengenali karakter, sebaliknya, OCR akan sangat sukar bahkan gagal melakukan pendeteksian(Eikvil,1993).

Pada umumnya sistem OCR terdiri dari beberapa komponen yang dapat dilihat pada gambar 3.1.



Gambar 3.1 *Flowchart* Tahapan Umum Pada Sistem OCR (Jain et al, 2013)

Preprocessing atau pemrosesan awal dilakukan terhadap gambar bertujuan menghasilkan data yang mudah dikenali sistem OCR secara akurat(Jain et al, 2013). Secara sederhana, *preprocessing* akan menghilangkan objek yang tidak diinginkan pada gambar (bukan karakter yang akan dikenali) dan untuk memperbaiki kualitas gambar(Aprilia, 2012). Perbaikan mutu citra tersebut meliputi memisahkan teks dengan latar belakang (hitam diatas putih), penyamaan ukuran seperti 12 pt. 300dpi, penajaman karakter (Jain et al, 2013).

Secara umum *Preprocessing* yang dilakukan terhadap citra diantaranya *grayscaling*, menghilangkan cacat(*noise removal*), dan *thresholding*. *Grayscaling* adalah tahap awal image preprocessing, yaitu mengubah gambar berwarna menjadi gambar yang hanya memiliki derajat keabuan. Selanjutnya, dilakukan *noise filtering*, yakni proses mengurangi cacat(*noise*). Proses

akhir dari *image preprocessing* adalah *thresholding* yang berguna memisahkan objek yang diamati dengan latar belakangnya dengan cara mengubah gambar menjadi hitam putih(Aprilia, 2012).

Menurut Jain, dkk., *Preprocessing* citra dapat melibatkan satu atau lebih dari langkah berikut (Jain et al, 2013) :

1. Reduksi *noise*

Dibandingkan dengan pemindai optik(*opical scanner*), sifat kamera digital lebih cenderung menangkap *noise* karena mode operasinya. Sensor citra pada kamera digital *charge-coupled device* (CCD) dan atau *Complementary Metal Oxide Semiconductor* (CMOS) tidak mampu menangani keadaan gelap dan *noise* yang tertangkap. Dalam keadaan pencahayaan yang sedikit atau ketika sensor menerima sedikit cahaya disebabkan bukaan lensa lebih kecil akan mengakibatkan lebih banyak *noise* yang terdeteksi. Jika menggunakan kamera dengan resolusi rendah, kemungkinan garis karakter yang muncul akan tipis dan menyatu dengan latar belakang. Bila teknik binerisasi dilakukan pada gambar tersebut kemungkinan besar garis-garis karakter tidak dikenali sebagai bagian latar depan(*foreground*). Masalah tersebut dapat diatasi dengan reduksi *noise* dan perbaikan mutu citra sebelum binerisasi citra.

2. *Skew* dan Perbaikan Perspektif (*perspective correction*)

Penyimpangan perspektif pada citra yang diambil dari kamera tidak dapat dihindari dan perbaikan perspektif sangat dibutuhkan sebab masalah ini memberi pengaruh

langsung terhadap kehandalan dan efisiensi proses segmentasi dan ekstraksi fitur.

3. Binerisasi

Sebuah proses konversi citra berwarna atau citra *grayscale* ke suatu citra *bi-level*. Setiap piksel dikategorikan sebagai salah satu latar depan atau latar belakang(Sezgin, 2004). *Thresholding* adalah metode binerisasi citra yang paling populer dan diimplementasikan melalui 2 cara berikut :

Global yakni diterapkan pada citra secara menyeluruh. Setiap piksel dikategorikan menggunakan karakteristik citra yang umum/global. Algoritma *Thresholding* Otsu global sering menjadi pilihan dalam hal ini.

Local Adaptive Thresholding yaitu nilai *threshold* yang berbeda untuk setiap piksel citra diperkirakan berdasar karakteristik lokal.

Pada tahap ekstraksi fitur, setiap karakter diwakili vektor fitur yang akan menjadi identitas karakter tersebut. Dalam proses pelatihan mesin OCR, vektor fitur dari karakter-karakter dipersiapkan sebagai *template* untuk digunakan pada tahap pencocokan fitur(*matching feature*) dari simbol-simbol pada sebuah citra. Metode ekstraksi fitur menganalisis input citra dan menyeleksi satu set fitur yang secara unik mengidentifikasi dan mengklasifikasikan karakter. Tujuan dari ekstraksi fitur adalah mengekstrak satu set

fitur yang akan memaksimalkan proses pengenalan(Jain et al, 2013).

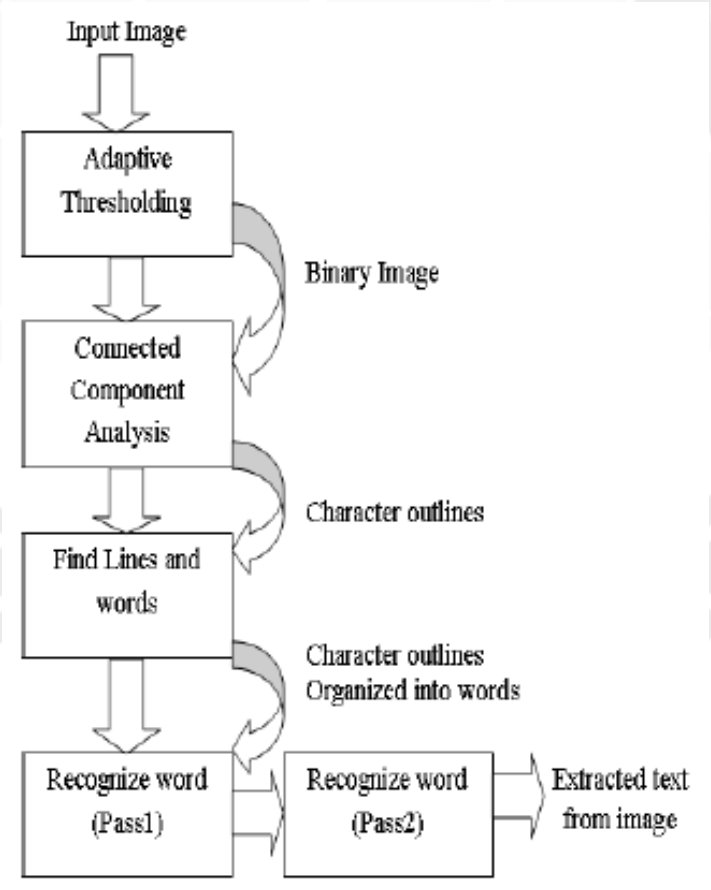
Tahap pengenalan(*Matching and Discrimination*) melakukan konversi citra dari setiap karakter ke dalam kode karakter yang sesuai dengan cara menyesuaikan/mencocokkan fitur karakter yang diperoleh dari proses ekstraksi fitur dengan fitur-fitur dari *template* karakter dan hasilnya dipisahkan ke dalam kode karakter tertentu. Algoritma pengenalan dapat menghasilkan keluaran yang bervariasi dari suatu citra dan menyediakan analisis probabilitas yang sama. Contoh, pengenalan citra dari karakter "1" dapat menghasilkan "1", "1" "/", "\" kode dan mencatat probabilitas dari setiap kejadian(Jain et al, 2013).

3.2 Tesseract

Google Tesseract merupakan mesin *Optical Character Recognition*(OCR) untuk berbagai sistem operasi yang awalnya dikembangkan oleh *Hewlet Packard* pada tahun 1985 hingga 1995(Smith, 2007). Tesseract dimasukkan dalam test akurasi OCR tahunan UNLV keempat sebagai "OCR Lab HP" tetapi kode-nya telah banyak berubah sejak saat itu, termasuk konversi ke *Unicode* dan *retraining*(latihan ulang) (Smith, 2007). Setelah tahun 2006, Tesseract dilepas oleh HP untuk digunakan secara bebas. Sejak saat itu, tesseract dikembangkan secara luas oleh Google dan dirilis di bawah lisensi Apache 2.0 serta dapat diakses di http://code.google.com/p/tesseract-ocr (Smith, 2007). Tesseract mungkin menjadi mesin OCR paling akurat saat ini. Tesseract dikombinasikan dengan *Leptonica Image*

Processing Library yang dapat membaca berbagai format gambar dan mengkonversikannya ke teks di lebih dari 60 bahasa. Dalam konferensi ICDAR, Smith mengungkapkan Tesseract mengkombinasikan kedua algoritma *matrix matching* dan ekstraksi fitur. Tesseract hanya memerlukan sedikit data pelatihan dan menggunakan kedua *static classifier* dan *adaptive classifier* sehingga memungkinkan mesin dapat berlatih sendiri pada dokumen yang dianalisis(Smith, 2007).

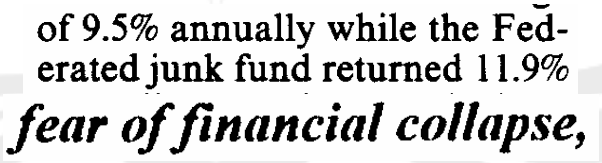
3.2.1 Arsitektur Tesseract



Gambar 3.2 Arsitektur Mesin OCR Tesseract(Patel et al, 2012)

Gambar 3.2 menunjukkan arsitektur Tesseract yang secara sederhana dapat dijelaskan sebagai berikut:

Tesseract OCR bekerja tahap demi tahap sebagaimana yang terlihat pada diagram pada gambar 3.2. Tahap pertama merupakan proses *adaptive thresholding* yakni konversi citra ke dalam citra biner. Langkah selanjutnya adalah analisis komponen terhubung/*Connected Component*(CC) yang berfungsi mengekstrak *outline* karakter. Metode ini memungkinkan pemrosesan OCR pada teks putih dengan latar belakang hitam. *Outline* yang ditemukan pada tahap ini dikonversi ke dalam "blob". *Blob* kemudian disusun ke baris teks, dan baris(lines) dan wilayah(*regions*) dianalisis untuk mengetahui apakah teks merupakan *pitch* tetap(*fixed pitch*) atau proporsional. Teks dengan pitch tetap(fixed-pitch) adalah teks yang setiap karakternya mempunyai ukuran ruang horizontal yang sama dalam suatu kata, berkebalikan dengan teks proporsional yang memiliki spasi kata yang tidak sama seperti pada gambar 3.3.



Gambar 3.3 Teks Non-*fixed-pitch*

Baris teks dipisah ke dalam kata-kata dan dibedakan menurut jenis spasi karakter. Jika teks adalah *fixed pitch* maka teks langsung dipecah dengan sel-sel karakter seperti pada gambar 3.4.

Gambar 3.4 Pemotongan Kata *Fixed-pitch*

Sebaliknya, untuk teks proporsional akan dipecah ke dalam kata menggunakan ruang pasti dan ruang fuzzy.

Pengenalan selanjutnya dilakukan melalui dua proses yang disebut *pass-two*(Smith, 2007). *Pass* pertama dilakukan untuk mengenali setiap kata secara bergilir. Keberhasilan *pass* pertama dipengaruhi oleh keberadaan kata dalam kamus dan tidak mempunyai makna ganda(ambigu), sehingga kata tersebut diteruskan ke *adaptive classifier* sebagai data pelatihan. *Adaptive classifier* kemudian dapat mengenali teks lebih akurat pada halaman selanjutnya. *Pass* kedua berfungsi melakukan pengenalan kembali kata-kata yang sukar dikenali atau terlewatkan pada *pass* pertama. Tahap akhir adalah pemrosesan kata-kata dengan ruang fuzzy dan memeriksa hipotesis alternatif pada ketinggian-x untuk mencari teks dengan *smallcap*(Smith, 2007).

3.2.2 Perbandingan Tesseract dengan Mesin OCR ABBYY FineReader

Dalam dokumentasi IMPACT(*Improving Access to Text*) yakni "Laporan Perbandingan mesin OCR Tesseract dan ABBYY FineReader" yang di publikasikan pada tahun 2012, menuliskan kesimpulan baik kelebihan maupun kekurangan yang ada pada kedua mesin OCR berdasarkan hasil beberapa tes yang dilakukan terhadap kedua mesin OCR(Helinski et al, 2012). Tes dilakukan terhadap mesin OCR Tesseract versi 3.01 dan ABBYY FineReader10 menghasilkan konklusi sebagai berikut(Helinski et al, 2012):

Tesseract berkemampuan lebih unggul pada kasus tes *gothic* dalam kebersihan atau kejelasan bentuk, dimana

sebagian besar karakter hieroglif(tulisan formal digunakan masyarakat mesir kuno) merupakan font fraktur(bentuk kaligrafi dari alfabet latin). Hal ini mungkin terjadi dikarenakan fakta bahwa pelatihan Tesseract hanya memasukkan glif fraktur, sedangkan ABBY hanya memasukkan pola glif modern maka klasifikasi glif lebih rentan terjadi kesalahan. Tesseract sukar mengatasi halaman(*page layout*) yang tidak beraturan dengan banyak cacat(*noise*). Dalam kasus batas halaman yang tidak beraturan, lebih banyak terdapat titik dan garis daripada suatu area yang solid, Tesseract memunculkan banyak kesalahan karakter, mengurangi presisi hasil dan nilai keakuratan. Hal ini jelas terlihat pada tes kasus *test-zr original antiqua* dimana keakuratan karakter hanya mencapai 39,4%. Pada observasi pelatihan *incremental* menunjukkan keakuratan pengenalan Tesseract dapat menurun karena adanya cacat halaman(Helinski et al, 2012).

FineReader memiliki kesulitan pada kasus tes dimana terdapat dua halaman yang telah melalui hasil OCR: halaman dengan *font* yang besar(±1600 karakter / halaman) dan halaman dengan *font* kecil(±2500 karakter / halaman). Keseluruhan hasilnya sangat buruk dikarenakan halaman dengan *font* kecil memberikan hasil yang sangat buruk(45% pada level karakter), meskipun halaman dengan *font* besar memberikan hasil yang relatif baik(76% pada level karakter) (Helinski et al, 2012).

Dari seluruh tes kasus dokumen *gothic* dan *antiqua* oleh Tesseract memberikan hasil buruk dikarenakan faktor berbagai tipe dokumen dan font, tata letak dan kejelasan cacat(*noise*) yang muncul pada halaman hasil

pemindaian(*scanned pages*). Meskipun demikian, dapat dinilai bahwa dalam kasus kualitas halaman yang baik, Tesseract memberikan hasil yang lebih baik dibandingkan FineReader(Helinski et al, 2012).

3.3 Android Text-To-Speech(TTS)

TTS (*Text to Speech*) merupakan salah satu aplikasi dalam bidang teknologi bahasa yang dapat mengkonversi teks dalam format suatu bahasa menjadi ucapan sesuai dengan pelafalan teks dalam bahasa yang digunakan, dengan cara melakukan fonetisasi, yaitu penyusunan fonem-fonem untuk membentuk ucapan. Sistem TTS pada umumnya melakukan dua proses konversi, yaitu konversi teks ke fonem dan konversi fonem ke ucapan. Kedua proses ini dilakukan berurutan dengan input berupa teks dan menghasilkan keluaran berupa suara atau ucapan(Arman, 2002).

3.4 Mesin Penerjemah Microsoft(Bing translate API)

Microsoft translator memberikan seperangkat layanan web API yang dapat digunakan oleh pengembang di dalam aplikasi, servis atau *web site*. API yang disediakan dapat digunakan dengan berbagai cara seperti HTTP REST servis, layanan AJAX, dan SOAP web servis. Microsoft Translator API tersedia di Windows Azure Marketplace, dijual sebagai berlangganan bulanan berdasarkan jumlah karakter teks yang dilewatkan ke API. API tersedia gratis untuk penggunaan karakter 2 juta per bulan.

Sampel metode yang disediakan API kepada pengembang:

1. *Translate* — menerjemahkan teks dari satu bahasa ke bahasa lain
2. *TranslateArray* — menerjemahkan array teks dari satu bahasa ke bahasa lain
3. *Speak (Text to Speech)* — menghasilkan bentuk audio dari suatu teks dalam bahasa tertentu
4. *Detect* — deteksi bahasa dari suatu String
5. *Detect Array* — deteksi bahasa dari suatu array String
6. *GetTranslations* — mengembalikan sebuah array dari alternatif terjemahan untuk teks
7. *GetTranslationsArray* — Mengembalikan array dari alternatif terjemahan untuk array teks
8. *GetLanguageNames*,*GetLanguagesForSpeak*, *GetLanguagesForTranslate* — mengembalikan sebuah daftar nama-nama bahasa, daftar audio bahasa yang tersedia, dan daftar kode-kode bahasa
9. *AddTranslation*, *AddTranslationArray* — mengizinkan pengembang menambahkan terjemahannya sendiri ke dalam infrastruktur Microsoft Translator.

3.5 Korean Pop(K-Pop)

Korean Pop atau *K-Pop* adalah salah satu fenomena musik dunia yang berasal dari Korea Selatan. *K-pop* merupakan bagian dari budaya pop modern Korsel atau sub-kultur yang terdiri dari beragam elemen seni dan hiburan Korea Selatan. *K-pop* memiliki ciri khas genre musik pop atau R&B Amerika dengan nuansa asia timur yang dipertunjukan oleh artis solo maupun yang paling berpengaruh yakni artis yang tergabung dalam *boyband* atau *girlband* dimana konsep grup musik ini lebih

mengutamakan keunggulan fisik dan kekompakan menari(Sulistyo, 2013).

Budaya pop Korea yang marak di Indonesia pada mulanya ditujukan untuk menyaingi impor budaya luar ke dalam Korea serta menambah pendapatan ekonomi negara, namun karena pasar Asia ternyata potensial dan sejalan dengan pertumbuhan ekonomi negara-negara di Asia, maka penyebaran budaya pop Korea ini menjadi sarana untuk melanggengkan kapitalisme Korea. Dengan semakin banyaknya penikmat budaya pop korea, maka akan memberikan dampak yang signifikan terhadap pertumbuhan ekonomi di Korea sendiri. Hal inilah yang dimanfaatkan kapitalis untuk memproduksi budaya pop Korea secara massal di berbagai wilayah Asia termasuk Indonesia(Sari et al, 2011).

Secara mudah bisa digambarkan bahwa fenomena *K-pop* secara tidak langsung semakin membuat bahasa Korea semakin terkenal. Semua produk budaya 'modern' Korea tersebut adalah asli Korea dalam arti baik lagu, sinetron, maupun filmnya memakai medium bahasa Korea sebagai bahasa pengantarnya. Sudah dipastikan ketika produk budaya tersebut memasuki negara-negara lain dan dikonsumsi oleh para konsumen di belahan negara lain, produk tersebut banyak yang masih menggunakan bahasa Korea—sebelum melalui proses sulih suara—terutama untuk drama. Di sinilah letak salah satu keberhasilan bangsa Korea dalam memperkenalkan bahasanya(Nugroho, 2011). Fakta bahwa bahasa Korea berada di posisi ke-10 sebagai bahasa yang sering digunakan di media internet pada tahun 2009(*Internet World Stats*, 2010). Hal ini menjadikan bahasa Korea sebagai salah satu bahasa yang

penetrasi dan signifikansinya di ajang global menjadi tak terbantahkan. Sebagai gambaran, bahasa Korea menempati urutan ke-13 dengan pengguna sebanyak 71 juta orang di dunia ini sebagai bahasa yang paling banyak digunakan. Jumlah ini pun baru dihitung dari Korea Selatan dan Korea Utara, belum termasuk para imigran Korea beserta keturunannya yang sampai saat ini berdiam di Cina, Amerika, Jepang, Rusia, Kanada, Australia, Amerika Selatan, Selandia Baru, Australia, dan negara-negara Eropa serta Asia lainnya(Nugroho, 2011).

3.6 Android

Android adalah susunan dari beberapa perangkat lunak (*software stack*). Stack ini secara umum meliputi sistem operasi, *middleware*, dan aplikasi-aplikasi kunci(Mulyana, 2012). Android pada awalnya tidak dikembangkan oleh google, melainkan dikembangkan oleh sebuah perusahaan bernama *Android Inc*. Karena google melihat banyaknya user yang *online* dengan perangkat *mobile*, maka google mengira bahwa perangkat mobile ini memiliki masa depan yang cerah, sehingga *Android Inc* diakuisi oleh Google pada tahun 2005. Beberapa hal penting seputar android (Winarno dan Zaki, 2011):

a. Android adalah sistem operasi *embedded* yang sangat bergantung pada kernel linux untuk layanan-layanan intinya, tapi Android bukanlah linux *embedded*.
b. Penulisan program untuk android menggunakan *framework* java, tapi ini bukanlah java, karena *library* standar java seperti *Swing* tidak didukung. *Library* lain seperti *timer* tidak disarankan, karena sudah diganti dengan *library default* dari android,

yang dioptimalkan untuk penggunaan di lingkungan *embedded* yang terbatas.

c. OS android merupakan sistem operasi *open source*, artinya developer bisa melihat semua *source code* sistem.

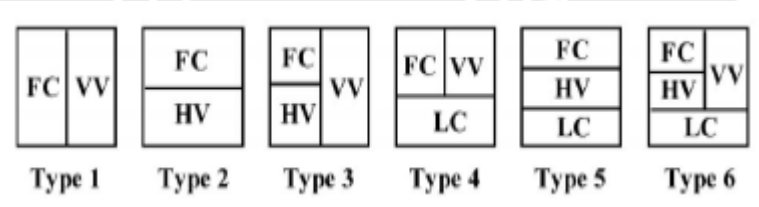
3.7 Eclipse IDE

Aplikasi android di tulis dan dibangun dengan menggunakan java, ada pula beberapa pilihan *Aplication Building Tools,* baik menggunakan IDE (*Intergrated Development Environment)* atau CLI (*Command line Interface).* Akan tetapi Google sangat mendukung Eclipse sebagai IDE java untuk mendukung aplikasi android dibandingkan dengan IDE yang lainnya, sebagai buktinya adalah dirilisnya plugin ADT untuk Eclipse. ADT (*Android Development Tools*) merupakan sebuah *plugin* untuk eclipse. Sebelum menggunakan atau menginstal ADT, pastikan bahwa eclipsenya sudah terinstal dan kompatibel dengan ADT dan android SDK. Android SDK (*Software Development Kit*) adalah *library* yang berisi kumpulan *tools*/alat bantu yang dibutuhkan dalam membangun/mengembangkan sebuah aplikasi di android dengan menggunakan bahasa pemrograman Java.

3.8 Karakter Korea

Bahasa Korea atau *hangeul* memiliki unsur vokal dan konsonan yang direpresentasikan dalam bentuk simbol-simbol. Masing-masing unsur vokal dan konsonan tidak dapat berdiri sendiri, sebab dalam membentuk sebuah kata diperlukan beberapa simbol seperti halnya bahasa Indonesia dimana satu kata tersusun dari gabungan

huruf-huruf. Pada umumnya, setiap karakter Korea terbentuk dari dua sampai empat simbol yang merupakan kombinasi vokal dan konsonan(Sofyan, 2010,h3; Yeung et al, 2006, h432) yang diatur pada ruang dua dimensi(Yeung et al, 2006; Jung dan Kim, 2000). Gambar 3.3 menunjukkan struktur karakter Korea(Jung dan Kim, 2000).



Gambar 3.5 Enam Jenis Aturan Komposisi Grafem Bahasa Korea(Aprilia, 2012)

grapheme type	graphemes
first consonant (FC)	ㄱ ㄴ ㄷ ㄹ ㅁ ㅂ ㅅ ㅇ ㅈ ㅊ ㅋ ㅌ ㅍ ㅎ ㄲ ㄸ ㅃ ㅆ ㅉ
last consonant (LC)	ㄱ ㄴ ㄷ ㄹ ㅁ ㅂ ㅅ ㅇ ㅈ ㅊ ㅋ ㅌ ㅍ ㅎ ㄲ ㅆ ㄳ ㄵ ㄶ ㄺ ㄻ ㄼ ㄽ ㄾ ㄿ ㅀ ㅄ
horizontal vowel (HV)	ㅗ ㅛ ㅜ ㅠ ㅡ
vertical vowel (VV)	ㅏ ㅑ ㅓ ㅕ ㅣ ㅐ ㅒ ㅔ ㅖ

Gambar 3.6 Grafem Bahasa Korea(Aprilia, 2012)

Abjad Korea terdiri dari 14 konsonan tunggal, 10 vokal tunggal, 5 konsonan rangkap, 11 vokal rangkap, 7 bunyi konsonan akhir, dan 11 konsonan akhir kompleks(Putra, 2011). Jumlah karakter yang dapat dibentuk dari kombinasi seluruh konsonan dan vokal sebanyak 11.728 karakter, tetapi pada prakteknya hanya sekitar 2.350 karakter yang digunakan dan umumnya hanya 520 karakter yang sering digunakan(Yeung et al, 2006). Berikut merupakan daftar simbol konsonan dan vokal yang ditampilkan dalam tabel 3.1, tabel 3.2, dan tabel 3.3.

Tabel 3.1 Daftar Vokal Tunggal dan Rangkap dalam Abjad Korea

Vokal Tunggal		Vokal Rangkap	
Huruf	**Nama**	**Huruf**	**Nama**
ㅏ	a (아)	ㅐ	ae (애)
ㅑ	ya (야)	ㅒ	yae (얘)
ㅓ	eo (어)	ㅔ	e (에)
ㅕ	yeo (여)	ㅖ	ye (예)
ㅗ	o (오)	ㅘ	wa (와)
ㅛ	yo (요)	ㅙ	wae (왜)
ㅜ	u (우)	ㅚ	oe (외)
ㅠ	yu (유)	ㅝ	wo (워)
ㅡ	eu (으)	ㅞ	we (웨)
ㅣ	i (이)	ㅟ	wi (위)
		ㅢ	ui (의)

Tabel 3.2 Daftar Konsonan Tunggal dalam Abjad Korea

Huruf	Nama	Pelafalan		
		Awal	**Tengah/akhir**	**Bawah**
ㄱ	Giuk	G/K	G	K
ㄴ	Niun	N	N	N
ㄷ	Digut	D/T	D	T
ㄹ	Riul	R/L	R	L
ㅁ	Mium	M	M	M
ㅂ	Biup	B	B	P
ㅅ	Siot	S	S	T
ㅇ	Iung	-	-	NG
ㅈ	jhiut	JH	JH	T

ㅊ	Chiut	CH	CH	T
ㅋ	Khiuk	KH	KH	K
ㅌ	Thiut	TH	TH	T
ㅍ	Phiup	PH	PH	P
ㅎ	Hiut	H	H	T

Tabel 3.3 Daftar Konsonan rangkap dalam Abjad Korea

Huruf	**Nama**	**Pelafalan**		
		Awal	**Tengah/akhir**	**Bawah**
ㄲ	ssanggiyeok (쌍기역)	K	K	K
ㄸ	ssangdigeut (쌍디귿)	T	T	-
ㅃ	ssangbieup (쌍비읍)	P	P	-
ㅆ	ssangsiot (쌍시옷)	S	S	T
ㅉ	ssangjieut (쌍지읒)	C	C	-

Dalam bahasa Korea, sistem penulisan menggabungkan huruf vokal dan konsonan. Setiap penulisan huruf vokal tunggal maupun ganda harus ditambahkan simbol "ㅇ"(ieung) (Aprilia, 2012). Simbol tersebut berbunyi "ng" seperti pada kata "uang", tetapi pada penulisan vokal bunyi "ng" tidak dibaca(Aprilia, 2012). Untuk mendapat huruf "a" diperlukan dua huruf "ㅇ" (ieung) dan "ㅏ" (a) sehingga menjadi "아" yang dibaca "a" seperti dalam bahasa Indonesia(Aprilia, 2012). Berbeda pada penulisan huruf konsonan tidak perlu ditambah huruf "ㅇ" (ieung)(Aprilia, 2012).

Pada penggabungan huruf vokal dan konsonan, posisi "ㅇ" (ieung) diganti oleh huruf konsonan(Sofyan, 2010, h2) jika huruf konsonan berada di depan huruf vokal

seperti pada kata “da”, sehingga penulisannya menjadi “다”. Sebaliknya, jika huruf konsonan terletak di belakang huruf vokal seperti pada kata “il” maka huruf konsonan tidak mengganti posisi “ㅇ” (ieung), tetapi ditulis dibawah huruf vokal, sehingga penulisannya menjadi vokal “이” (i) ditambah konsonan “ㄹ” dibawahnya yang akan menghasilkan “일” (il) (Aprilia, 2012). Pada contoh kata “sam” penggabungannya terdiri dari huruf konsonan “s” mengganti huruf “ㅇ” (ieung) dari huruf vokal “a” kemudian huruf konsonan “m” ditulis dibawah huruf gabungan “s” dan “a” (Aprilia, 2012).

Pada bab landasan teori telah dipaparkan teori-teori dasar yang digunakan oleh penulis dalam pengembangan perangkat lunak. Pada bab selanjutnya, bab analisis dan perancangan perangkat lunak akan dibahas mengenai analisis yang dilakukan penulis dalam pengembangan perangkat lunak.