BAB 2
TINJAUAN PUSTAKA

2.1 Citra

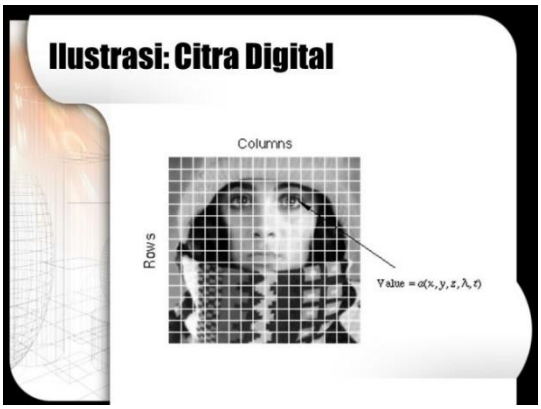
Citra adalah suatu representasi (gambaran), kemiripan, atau imitasi dari suatu objek. Citra terbagi 2 yaitu citra yang bersifat analog dan ada citra yang bersifat digital. Citra analog adalah citra yang bersifat *continue* seperti gambar pada monitor televisi, foto sinar X, dan lain-lain. Sedangkan pada citra digital adalah citra yang dapat diolah oleh computer [7].

2.1.1 Citra Analog

Citra analog adalah citra yang bersifat *continue*, seperti gambar pada monitor televisi, foto sinar X, foto yang tercetak di kertas foto, lukisan, pemandangan alam, hasil CT scan, gambar-gambar yang terekam pada pita kaset, dan lain sebagainya. Citra analog tidak dapat direpresentasikan dalam komputer, sehingga tidak bisa diproses di komputer secara langsung [7].

2.1.2 Citra Digital

Citra digital merupakan representatif dari citra yang diambil oleh mesin dengan bentuk pendekatan berdasarkan sampling dan kuantisasi. Sampling menyatakan besarnya kotak-kotak yang disusun dalam baris dan kolom. Dengan kata lain, sampling pada citra menyatakan besar kecilnya ukuran *pixel* (titik) pada citra, dan kuantisasi menyatakan besarnya nilai tingkat kecerahan yang dinyatakan dalam nilai tingkat keabuan (*grayscale*) sesuai dengan jurnlah bit biner yang digunakan oleh mesin, dengan kata lain kuantisasi pada citra menyatakan jumlah warna yang ada pada citra. [7] Pada gambar 2.1 merupakan contoh ilustrasi digital citra.



Gambar 2. 1 Ilustrasi Digitalisasi Citra

2.1.3 Pengolahan citra Digital

Pengolahan citra *(image Processing)* merupakan proses mengolah piksel-piksel di dalam citra digital untuk tujuan tertentu. Pada awalnya pengolahan citra ini dilakukan untuk memperbaiki kualitas citra, namun dengan berkembangnya dunia komputasi yang ditandai dengan semakin meningkatnya kapasitas dan kecepatan proses komputer serta munculnya ilmu-ilmu komputasi yang memungkinkan manusia dapat mengambil informasi dari suatu citra [7].

Proses pengolahan citra secara diagram proses dimulai dari pengambilan citra, perbaikan kualitas citra, sampai dengan pernyataan representatif citra yang dicitrakan sebagai berikut:

Gambar 2. 2 Proses Pengolahan Citra

Dalam perkembangan lebih lanjut, *image processing* dan *computer vision* digunakan sebagai mata manusia, dengan perangkat *input image capture* seperti kamera dan scanner dijadikan sebagai mata dan mesin komputer (dengan program

komputasinya) dijadikan sebagai otak yang mengolah informasi. Sehingga muncul beberapa pecahan bidang yang menjadi penting dalam computer vision, antara lain: pattern recognition (pengenalan pola), biometric pengenalan identifikasi manusia berdasarkan ciri-ciri biologis yang tampak pada badan manusia), *content based image*, and *video retrieval* (mendapatkan kembali citra atau video dengan informasi tertentu), video editing, dan lain-lain [7].

2.2 Pengenalan pola

Pola adalah suatu entitas yang terdefinisi (mungkin secara samar) dan dapat diidentifikasi serta diberi nama. Pola bisa merupakan kumpulan hasil pengukuran atau pemantauan dan bisa dinyatakan dalam notasi vektor. Contoh : sidik jari, raut wajah, gelombang suara, tulisan tangan dan lain sebagainya. Dalam pengenalan pola data yang akan dikenali biasanya dalam bentuk citra atau gambar, akan tetapi ada pula yang berupa suara [8].

Secara umum pengenalan pola adalah suatu ilmu untuk mengklasifikasikan atau menggambarkan sesuatu berdasarkan pengukuran kuantitatif fitur (ciri) atau sifat utama dari suatu obyek.

Menurut Theodoridis dan Koutroumbasi [8], pengenalan pola *(pattern recognition)* dapat diartikan sebagai proses klasifikasi dari objek atau pola menjadi beberapa kategori atau kelas yang bertujuan untuk pengambilan keputusan.

2.1.1 Penerapan Pattern Recognition

Karena kompleksitas area AI maka dibuatkan sub-sub bagian yang dapat berdiri sendiri dan dapat berdiri sendiri saling bekerja sama dengan sub bagian lain atau dengan disiplin ilmu lain [8]. Berikut ini beberapa cabang ilmu sub bagian dari AI:

1. *Character Recognition*

   Salah satu era pengenalan pola yang secara umum menangani permasalahan otomatisasi dan informasi. Sistem OCR mempunyai front end device yang terdiri dari lensa scan, document, data transport dan sebuah detector.

2. *Speech Recognition*

Pengenalan pola yang secara umum telah banyak dikembangkan saat ini. Sistem ini mengijinkan kita untuk berkomunikasi antara manusia dengan memasukan data ke komputer. Meningkatkan efisiensi industri manufaktur, mengontrol mesin, dan sebagainya.

3. *Face Recognition*

Salah satu pengaplikasian sistem pengenalan pola yang ditunjukan kepada pengenalan pola wajah, face recognition adalah sebuah sistem yang mengenali image wajah manusia yang biasa digunakan untuk otomatisasi dan security sebuah industri.

4. *Machine Vision*

Sebuah sistem mesin yang menggunakan pengenalan pola sebagai dasar dari tahapannya. Mesin ini menangkap sebuah atau sekelompok objek dengan kamera dan selanjutnya dianalisa untuk di deskripsikan.

2.3 Aksara Latin (Abjad)

Pada umumnya tulisan semua Bahasa di Dunia ini ditulis dengan abjad atau secara *alphabet*, walaupun masih banyak lagi Bahasa yang menggunakan symbol-simbol tertentu dalam penulisannya. Dalam tugas akhir ini peneliti mengambil studi kasus yaitu aksara latin (abjad). Tabel 2.1 adalah contoh huruf Alphabet.

Aa	Bb	Cc	Dd	Ee

Tabel 2. 1 Alfabet Huruf

Ff	Gg	Hh	Ii	Jj
Kk	Ll	Mm	Nn	Oo
Pp	Qq	Rr	Ss	Tt
Uu	Vv	Ww	Xx	Yy
Zz				

2.4 *Grayscale*

Citra grayscale menangani gradasi warna hitam dan putih, yang menghasilkan efek warna abu-abu, warna gambar dinyatakan dengan intensitas. Dalam hal ini, intensitas berkisar antara 0 sampai dengan 255. Nilai 0 menyatakan hitam dan nilai 255 menyatakan putih Berikut adalah rumus konversi citra berwarna (RGB) menjadi grayscale.

$$I = (0.2989 * R) + (0.5870 * G) + (0.1141 * B) \qquad (2.1)$$

Keterangan :

I = fungsi pencarian nilai skala keabu-abuan

R = komponen nilai merah (*Red)* dari suatu titik *pixel*

G = komponen nilai hijau (*Green*) dari suatu titik *pixel*

B = komponen nilai biru (*Blue*) dari suatu titik *pixel*

Persamaan di atas merupakan salah satu rumus yang digunakan untuk mengkonversikan citra berwarna menjadi grayscale. Persamaan 2.1 dipilih dalam

penelitian ini karena mata manusia seacara alami sensitif terhadap cahaya merah dan hijau. Maka dari itu, warna-warna ini diberi bobot yang lebih tinggi untuk memastikan bahwa keseimbangan intesitas relatif dalam citra grayscale yang dihasilkan mirip dengan citra warna RGB [7].

2.5 *Sauvola threshold*

Sauvola threshold merupakan metode *threshold* yang mampu mendeteksi citra tulisan tangan dengan sangat baik, dengan cara menghitung ambang menggunakan rentang nilai dinamis dari nilai standar deviasi citra *grayscale* [9]. *Sauvola* termasuk dalam *local threshold* dimana nilai ambang ditentukan oleh nilai pixel tetangga, tujuan dilakukannya proses *threshold* adalah untuk menyederhanakan bentuk citra.

$$T(x,y) = m(m,y)n * (1 + k * \left(\frac{s(x,y)}{R} - 1\right)) \quad (2.2)$$

Pada rumus (2.2) terdapat rumus unutk menghitung m(x,y) yang dapat dihitung menggunakan rumus (2.3) dan rumus unutk menghitung s(x,y) dapat dilihat pada rumus (2.4)

$$m(x,y) = \frac{\sum_{i=min}^{i=max} \sum_{j=min}^{j=max} img(i,j)}{i*j} \quad (2.3)$$

$$s(x,y) = \frac{\sum_{i=min}^{i=max} \sum_{j=min}^{j=max} (img(i,j) - m(x,y))^2}{(i*j)-1} \quad (2.4)$$

Keterangan :

R : nilai maksimum dari standar deviasi (128 untuk citra grayscale)

k : kernel dengan nilai antara 0.2 – 0.5

m : fungsi yang menghasilkan nilai rata-rata dari sejumlah pixel citra.

s : fungsi yang menghasilkan deviasi dari sejumlah pixel citra

T : fungsi yang menghasilkan nilai threshold (ambang)

x : nilai koordinat lebar citra dalam rumus (i)

y : nilai koordinat tinggi citra dalam rumus (j)

Setelah nilai ambang T(x,y) sudah didaptkan, selanjutnya masukkan ke persamaan (2.5) di bawah ini [9].

$$f(x,y) = \begin{cases} 0, img(x,y) < T(x,y) \\ 255, img(x,y) \geq T(x,y) \end{cases} \quad (2.5)$$

Keterangan :

f: fungsi yang menghasilkan nilai 0 atau 255

img: nilai grayscale citra

2.7 Segmentasi Citra

Segmentasi citra adalah pemisahan objek yang satu dengan objek yang lain dalam suatu citra atau antara objek dengan latar yang terdapat dalam sebuah citra. Dengan proses segmentasi tersebut, masing-masing objek pada citra dapat diambil secara individu sehingga dapat digunakan sebagi input bagi proses lain. Ada 2 macam segmentasi, yaitu *full segmentation* dan *partial segmentation*. *Full segmentation* adalah pemisahan suatu object secara individu dari *background* dan diberi ID (label) pada tiap-tiap segmen. *Partial segmentation* adalah pemisahan sejumlah data dari background dimana data yang disimpan hanya data yang dipisahkan saja untuk mempercepat proses selanjutnya [10].

2.8 Ekstrasi Ciri

Ekstrasi fitur adalah adalah proses pengukuran terhadap data yang telah di normalisasi untuk membentuk sebuah fitur. Nilai fitur digunakan oleh pengklasifikasi

untuk mengenali unit masukan dengan unit target keluaran dan memudahkan pengklasifikasian karena nilai ini mudah untuk dibedakan [11].

Pada penelitian ini, penulis menggunakan metode ekstraksi *Directional element feature*.karena telah banyak digunakan dan terbukti memberikan hasil yang baik dalam pengenalan tulisan tangan

2.8.1 Ekstrasi ciri DEF

Directional Element Feature merupakan metode ekstrasi ciri yang telah banyak digunakan dan terbukti memberikan hasil yang baik dalam pengenalan tulisan tangan. Salah satunya adalah yang pernah diterapkan untuk huruf cina [3]. Metode DEF pada OCR yang melihat perbedaan kontur dan tanpa mengalami proses *skeletonizing* [12]

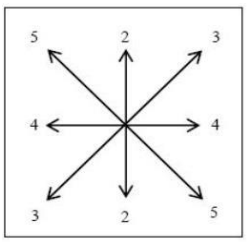
Berikut adalah tahapan-tahapan yang ada dalam ekstrasi DEF [12]:

1. ***Image Scaling***

   Citra akan di-*scaling* sehingga ukurunnya menjadi seragam *NxN pixel*, hal ini dilakukan supaya semua citra yang telah disegmentasi mempunyai ukuran yang sama untuk mendapatkan ciri yang baik.

2. ***Dot Orientation***

   Directional Element Feature adalah pencarian nilai *feature* berdasarkan label arah dari sebuah pixel. Pada metode ini setiap pixel f*oreground*(piksel karakter) pada citra digital akan diberikan label (*feature* arah) dengan cara menelusuri tetangga dari masing-masing pixel *foreground* searah perputaran jarum jam dimana arah yang digunakan terdiri dari 4 arah. Besar nilai arah yang digunakan bisa dilihat pada gambar Contoh pembagi 3 zona pada citra biner dapat dilihat pada gambar [3] 2.3.



Gambar 2. 3 Nilai Arah

Untuk melakukan pelabelan arah pada masing-masing *pixel* dapat dilakukan dengan cara sebagai berikut [3]:

1. Lakukan pengecekan secara raster dari kiri ke kanan
2. Apabila menemukan sebuah *pixel foreground* maka lakukan pengecekan dengan melihat tetangga dari *pixel* tersebut.
3. Perhatikan gambar 2.4 Misalkan 0 adalah sebuah *foreground*, maka nilai arah 0 didapatkan dengan melakukan pengecekan secara berurutan dari X1-X8. pengecekan berhenti ketika menemukan X yang merupakan *foreground*, maka ubahlah nilai 0 menjadi nilai arah berdasarkan aturan di bawah ini:
    a) Jika posisi X1 atau X5 maka nilainya 5
    b) Jika posisi X2 atau X6 maka nilainya 2
    c) Jika posisi X3 atau X7 maka nilainya 3
    d) Jika posisi X4 atau X8 maka nilainya 4

X1	X2	X3
X8	O	X4
X7	X6	X5

Gambar 2. 4 Nilai Arah DEF

3. Vektor Construction

Tahapan ini untuk menghitung nilai dari setiap *pixel* ketetanggaan yang sebelumnya telah melalui proses *dot orientation,* awalnya citra akan dibagi

menjadi (piksel)/$N_1 \times N_1$ area yang saling overlap sebanyak $N_1/2$ *pixel* terhadap area yang bertetanggaan.

Selanjutnya area, setiap area dibagi kembali menjadi 4 *sub-area,* yaitu A, B, C dan D, yang eksklusif satu sama lainnya. Setelah diperoleh jumlah pixel di setiap *sub-area,* akan dikalikan masing-masing jumlah elemen ketetanggan disetiap *sub-area* dengan bobot(*w*) pada *sub-area* tersebut dan dijumlahkan dari setiap elemen di *sub-area* dengan rumus sebagai berikut [3]:

$$x_j = w_A x_j + w_B x_j + w_C x_j + w_D x_j$$
$$j = 1, \ldots, 4$$

Dimana j merupakan pixel ketetanggaan.

Selanjutnya vektor ciri DEF yang diperolh dengan menggabungkan jumlah elemen disetiap area menjadi satu kolom vektor *V.*

$$V = [X_1^1, X_2^1, X_3^1, X_4^1, \ldots, X_1^N, X_2^N, X_3^N, X_4^N]$$

Dimana *N* merupakan jumlah area dari setiap karakter. Vektor *V* inilah yang nanntinya jadi input untuk proses *learning* dan *testing.*

2.9 Teknik Learning (Belajar)

Teknik *learning* merupakan salah satu bagian dari ilmu kecerdasan buatan. Dalam learning kita tidak harus tahu anturan yang berlaku dalam sistem yang dibuat, melainkan aturan yang diharapkan bisa diharapkan secara otomatis ditemukan. Proses belajar dalam teknik learning menggunakan data-data masukan sebagai pengalaman yang baru untuk dapat meningkatkan performasi dari sistem. Program komputer yang sanggup belajar adalah program yang bisa meningkatkan performasinya melalui pengalaman [11].Ada beberapa metode yang menggunakan teknik *learning* seperti *decision tree,* jaringan saraf tiruan, algoritma genetika dan lain-lain.

2.10 Metode Min-Max Normalization

Min-Max Normalization adalah sebuah metode normalisasi yang mentrasformasi data secara liniear menjadi sebuah range yang baru. Formula dari metode ini dapat dilihat pada persamaan [13]

$$v' = \frac{v' - \min A}{\max A - \min A} \quad (new\_\max_A - new\_\min_A) + new\_\min_A$$

2.11 Algoritma Viterbi

Algoritma Viterbi adalah algoritma dynamic programming untuk menemukan kemungkinan rangkaian status yang tersembunyi (biasa disebut Viterbi path) yang dihasilkan pada rangkaian pengamatan kejadian, terutama dalam lingkup HMM.

Untuk menemukan sebuha rangkaian status terbaik, q = ($q_1 q_{2\ldots.} q\boldsymbol{\tau}$), untuk rangakaian observasi O = ${}^{O}{}_1\,{}^{O}{}_2\ldots\ldots{}^{O}\boldsymbol{\tau}$), perlu didefinisikan kwantitas:

(1) $\boldsymbol{\delta}t(i) = \max P[q_1 q_2 \ldots q_{t-1}, q_t = {}_{I},{}^{o}1\ \ 02\ldots..ot \mid \boldsymbol{\lambda}\ ]$.

$\boldsymbol{\delta}$t (i) adalah rangkaian terbaik, yaitu dengan kemungkinan terbesar, pada waktu t dimana perhitungan untuk pengamatan t pertama dan berakhir pada status i. dengan menginduksi, didapat:

(2) ${}^{\delta}{}_{t+1}(j) = [\max {}^{\delta}{}_t(i)_{ij}]\ b_j(O_{1+1})$

Untuk mendapatkan kembali rangkaian status, perlu adanya penyimpanan hasil yang memaksimalkan persamaan (2), untuk tiap i dan j, dengan menggunkan $A_r(j)$,

Prosedur lengkap untuk menemukan kumpulan status-status terbaik bias dirumuskan

Sebagai [14].

1. Inisialisasi $\delta 1\ (i) = \Pi ibi(o_1)$, $1 \leq i \geq N\ A_r(1) = 0$.

$$Ar(1) = 0$$

2. Rekursif $\delta t(i) = \max \ [\delta_{t}\text{-}1(i)a_{ij}]b_j(o_t)$

$1 \leq i \leq N$

$2 \leq t \leq T, 1 \leq j \leq N$

$Ar(j) = \arg\max [^{\delta}{}_{t-1}(i)a_{ij}]$

$1 \leq i \leq N$

$2 \leq t \leq T, 1 \leq j \leq N$

3. Terminasi $P^* = \max [\delta_T(i)]$

$1 \leq i \leq N$

4 Lacak balik $= qt^* = Ar(t+1)(q^*_{t+1})$

$t = T\text{-}1, T\text{-}2, \ldots, 1.$

2.12 Hidden Markov Model

Hidden Markov Model (HMM) adalah pemodelan probabilitas suatu sistem dengan mencari parameter-parameter yang tidak diketahui untuk mempermudah proses analisis sistem tersebut [15]. *Hidden Markov Model* dapat digunakan untuk aplikasi dibidang *temporal pattern recognition* ‗pengenalan pola temporal‗ seperti pengenalan suara, tulisan, gestur, bioinformatika, kompresi kalimat, *computer vision*, ekonomi, finansial, dan pengenalan not balok.

HMM adalah variasi dari *finite state machine* yang memiliki kondisi tersembunyi *Q*, suatu nilai output *O* (observasi), kemungkinan transisi *A*, kemungkinan output *B*, sebuah kondisi awal π. Kondisi saat ini tidak terobservasi. Tetapi, setiap keadaan menghasilkan output kemungkinan *B*. biasanya, *Q* dam *O* dimengerti, jadi HMM disebut triple (*A*,*B*, π).

1 Himpunan observed state: $O = o_1, o_2, \ldots, oN.$

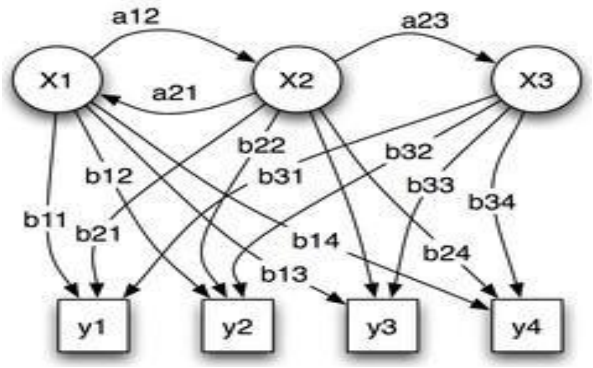
2 Himpunan hidden state: $Q = q_1, q_2, .. qN$

3 Probabilitas transisi: $A = a0_1, a0_2 \ldots, an_1$

4 …anm ; a_{ij} adalah probabilitas untuk pindah dari state i ke state j.

5 Probabilitas emisi atau observation likehood: $B = b_i(O_t)$, merupakan probabilitas observasi O_t dibangkitkan oleh state i.

6 State awal dan akhir: q_0, q_{end}, yang tidak terkait dengan observasi.



Gambar 2. 4 Representasi Parameter HMM

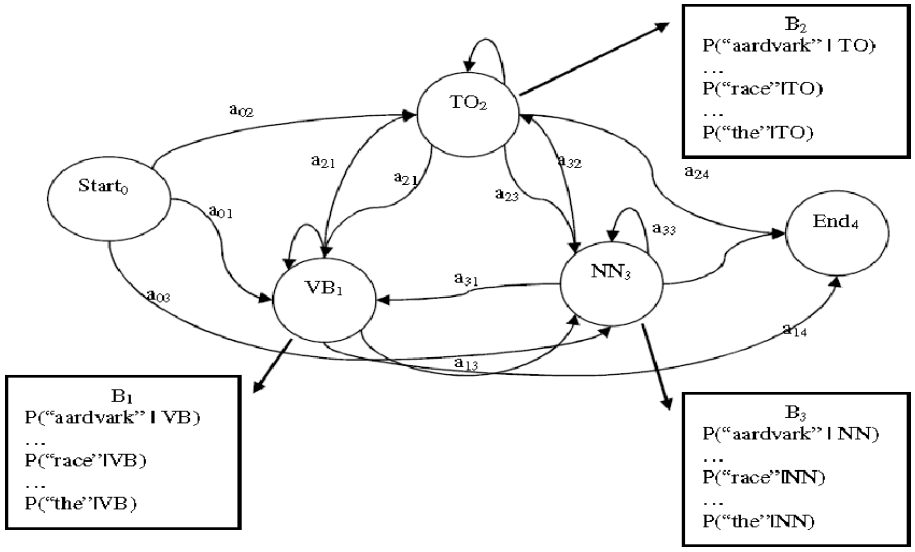
Penjelasan Gambar 2.5:

x = kondisi

y = observasi yang mungkin

a = kemungkinan keadaan transisi

b = kemungkinan output contoh kasus HMM



***Gambar 2. 5* Contoh HMM untuk part of speech tagger**

Pada gambar 2.5 kata yang dicari menggunakan metode HMM adalah ―aardvark‖, ―race‖, dan ―the‖. Ketiga kata tersebut menjadi *observed state*. Sedangkan hidden state adalah ―TO‖ (*to infinitive*), ―VB‖ *(verb base form*), dan ―NN‖ (mass noun). Bi himpunan semua probabilitas emisi untuk hidden state i, sedangkan aij adalah probabilitas transisi dari state i ke state j [16].

2.13 Pengujian Sistem

Pengujian sistem adalah proses pemeriksaan atau evaluasi sistem atau komponen sistem secara manual atau otomatis untuk memverifikasi apakah sistem memenuhi kebutuhan-kebutuhan yang dispesifikasikan atau mengidentifikasi perbedaan-perbedaan antara hasil yang diterapkan dengan hasil yang terjadi [17]. Pengujian seharusnya meliputi tiga konsep berikut:

1. Demonstrasi validasi perangkat lunak pada masing-masing tahap disiklus pengembangan sistem.
2. Penentuan validitas sistem akhir dikaitkan dengan kebutuhan pemakai.

3. Pemeriksaan perilaku sistem dengan mengeksekusi sistem pada data sampel pengujian.

Pada dasarnya pengujian diartikan sebagai aktivitas yang dapat atau hanya dilakukan setelah pengkodean (kode program selesai). Namun, pengujian seharusnya dilakukan dalam skala lebih luas. Pengujian dilakukan begitu spesifikasi kebutuhan telah dapat didefinisikan. Evaluasi terhadap spesifikasi dan perancangan juga merupakan teknik pengujian. Kategori pengujian dapat dikategorikan menjadi dua [17], yaitu :

1. Berdasarkan ketersediaan logik sistem, terdiri dari *Black box testing* dan *White box testing.*
2. Berdasarkan arah pengujian, terdiri dari pengujian *top down* dan Pengujian *bottom up.*

2.13.1 Pengujian Black Box

Konsep *black box* digunakan untuk mempresentasikan sistem yang cara kerja di dalamnya tidak tersedia untuk diinspeksi. Di dalam *black box*, item-item yang diuji dianggap "gelap" karena logiknya tidak diketahui, yang diketahui hanya apa yang masuk dan apa yang keluar dari *black box* [18].

1. *Graph-based testing*
2. *Equivalence partitioning*
3. *Comparison testing*
4. *Orthogonal array testing*

Pada pengujian *black box,* kita mencoba beragam masukan dan memeriksa kelauran yang dihasilkan. Kita dapat mempelajari apa yang dilakukan kotak, tapi tidak mengetahui sama sekali mengenai cara konversi dilakukan. Teknik pengujian *black box* juga dapat digunakan untuk pengujian berbasis skenario, dimana isi dalam sistem mungkin tidak tersedia untuk diinspeksi tapi masukan dan keluaran yang didefinisikan dengan *use case* dan informasi analisis yang lain.

2.14 Pengujian Akurasi

Akurasi merupakan ukuran ketepatan sistem dalam mengenal karakter tulisan tangan dan mencocokannya dengan data yang berada di database. Akusisi sistem secara matematis dapat dilakukan seperti pada persamaan pada (2.42) [18].

$$Akurasi = \frac{jumlah\ karakter\ sama}{jumlah\ seluruh\ karakter}\ X\ 100\% \qquad (2.42)$$

2.15 Bahasa pemograman

Bahasa pemograman diperlukan untuk menjalankan instruksi-instruksi apa yang harus dilakukan komputer. Komputer tidak bisa memahami bahasa manusia, sehingga diperlukan pengguna bahasa komputer di dalam program komputer. Penelitian ini menggunakan bahasa Java untuk mengembangkan aplikasi dan bahasa pemograman SQL untuk mengelola *database.*

2.15.1 Java

Bahasa Java dikembangkan oleh sebuah tim yang diketuai oleh James Gosling di Sun Microsystem. Java awalnya dikenal dengan Oak, yang didesain pada tahun 1991 untuk chip-chip yang tertanam pada peralatan-peralatan elektronik. Pada tahun 1995, diberi nama baru Java, yang didesain ulang untuk mengembangkan aplikasi-aplikasi internet. Java telah menjadi sangat populer. Perkembangannya yang sangat cepat dan penerimaannya di kalangan pengguna dapat dijejak dari karakteristik perancangannya, khususnya dari janji pengembang Java bahwa begitu anda menciptakan suatu program, maka anda bisa menjalankannya di mana saja [19].

Java memiliki banyak fitur, bahasa pemrograman bertujuan-umum yang dapat digunakan untuk mengembangkan aplikasi-aplikasi tingkat tinggi. Saat ini, Java tidak lagi hanya digunakan untuk pemrograman Web, tetapi juga aplikasi-aplikasi *standalone* bebas *platform* pada *server*, *desktop*, dan divais-divais bergerak (*mobile*) [19]. Bahasa Java juga telah digunakan untuk mengembangkan kode dalam berkomunikasi dan mengendalikan robot di Mars. Banyak perusahaan yang sebelumnya meremehkan keunggulan Java sekarang malah menggunakannya untuk

mengembangkan aplikasi-aplikasi terdistribusi yang dapat diakses oleh banyak konsumen melalui internet.

Java merupakan bahasa pemograman yang tangguh terbukti handal pada banyak aplikasi. Terdapat tiga edisi java [19], yaitu :

1. Java SE (*Standar Edition*)
   Digunakan untuk mengembangkan aplikasi-aplikasi pada sisi client atau *applet.*
2. Java EE (*Enterprise Edition*)
   Digunakan untuk mengembangkan aplikasi-aplikasi pada sisi *server,* seperti *Java Servlets* dan *Java server pages.*
3. Java ME (*Micro Edition*)
   Digunakan untuk mengembangkan aplikasi-aplikasi untuk *device* bergerak, seperti telepon genggam.

Peneletian ini menggunakan Java SE untuk mendapatkan aplikasi pengenalan tulisan tangan.

2.16 UML (*Unifide Modeling Language*)

UML *(Uniefied Modeling Language)* adalah notasi yang lengkap untuk membuat visualisasi model suatu sistem. Sistem berisi informasi dan fungsi, tetapi secara normal digunakan untuk memodelkan sistem komputer. Di dalam pemodelan objek guna menyajikan sistem yang berorientasi objek kepada orang lain, akan sangat sulit dilakukan dalam bentuk kode bahasa pemrograman [18].

UML disebut sebagai bahasa pemodelan bukan metode. Bahasa pemodelan (sebagian besar grafik) merupakan notasi model yang digunakan untuk mendesain secara cepat. Bahasa pemodelan merupakan bagian terpenting dari metode. UML merupakan bahasa standar untuk penulisan *blueprint software* yang digunakan untuk visualisasi, spesifikasi, pembentukan dan pendokumentasian.+*n* alat-alat dari sistem perangkat lunak. UML biasanya disajikan dalam bentuk diagram atau gambar yang

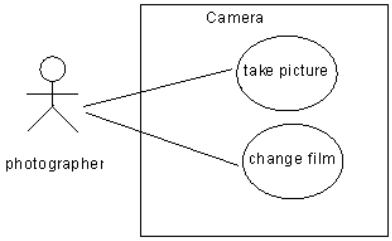
meliputi *class* beserta atribut dan operasinya, serta hubunganantar kelas. UML terdiri dari banyak diagram diantaranya *use case, diagram, activity diagram, class diagram, dan sequence diagram.*

2.16.1 Use Case Diagam

Dalam konteks UML, tahap konseptualisasi dilakukan dengan pembuatan *use case diagram* yang sesungguhnya merupakan deskripsi peringkat tinggi bagaimana perangkat lunak (aplikasi) akan digunakan oleh penggunannya.

Use case diagram merupakan deskripsi lengkap tentang interaksi yang terjadi antara para aktor dengan sistem [18]. Dalam hal ini, setiap objek yang berinteraksi dengan sistem merupakan aktor untuk sistem, sementara *use case* merupakan deskripsi lengkap tentang bagaimana sistem berprilaku kepada aktornya. Aktor dalam *use case diagram* digambarkan sebagai ikon yang berbentuk manusia, sementara *use case* digambarkan elips yang berisi nama *use case* yang bersangkutan. Untuk mempermudah pemahaman, aktor biasanya dituliskan sebagai kata benda, sementara *use case* biasanya ditulisakan dengan kata kerja.

berikut gambar 2.6 contoh Use Case Diagram



Gambar 2. 6 *Contoh Use Case Diagram*

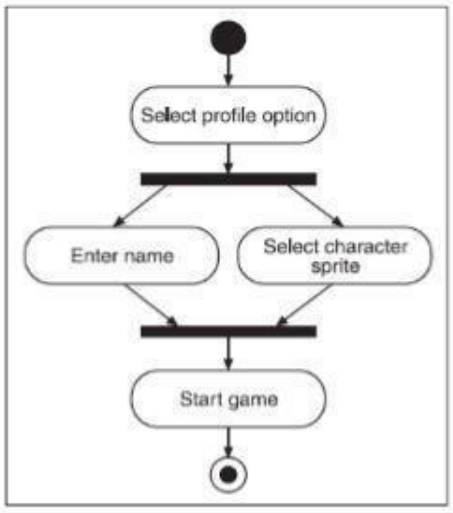
2.16.2 Activity Diagram

Activity Diagram adalah diagram *flowchart* yang diperluas yang menunjukkan aliran kendali satu aktivitas ke aktivitas lain. Kita menggunakan diagram ini memodelkan aspek dinamis sistem. Aktivitas adalah eksekusi notatomik yang berlangsung di *state machine*. *Activity diagram* mendeskripsikan aksi-aksi dan

hasilnya. *Activity diagram* berupa operasi-operasi dan aktivitas-aktivitas di *use case*. *Activity diagram* dapat digunakan untuk: [18]

1. Pandangan dalam yang dilakukan di operasi
2. Pandangan dalam bagaimanan objek-objek bekerja.
3. Pandangan dalam di aksi-aksi pada pengaruhnya pada objek-objek
4. Pandangan dalam dari suatu *use case*.
5. Logic dari proses bisnis.

Diagram aktivitas merupakan jenis khusus dan diagram *statechart*. State adalah aksi-aksi yang menuju state berikutnya setalah selesai aksi itu. Contoh activity diagram dapat dilihat pada gambar 2.7



Gambar 2. 7 Contoh Aktivity Diagram

Dari gambar 2.8 diatas merupakan contoh *activity diagram* dengan alur:

1. lingkaran hitam penuh menandakan memulai aktivitas
2. masuk ke proses fork unutk menjalankan aktivitas secaa pararel kemudian menjalankan aktivitas dibaeahnya

3. masuk ke proses join untuk menggabungkan proses yang dipisahkan oleh fork
4. melakukan aktivitas start game
5. lingkaran hitam sebagian menandakan akhir dari suatu aktivitas

2.16.3 Class Diagram

Class diagram merupakan diagram paling umum dipakai di semua pemodelan berorientasi objek. Pemodelan kelas merupakan pemodelan yang paling utama di pendekatan berorientasi objek. Pemodelan kelas menunjukkan kelas-kelas yang ada di sistem dan hubungan antar kelas-kelas itu, atribut-atribut dan operasi - operasi di setiap kelas [18]. Class diagram menunjuk aspek. statis sistem terutama untuk mendukung kebutuhan fungsional sistem. Meskipun class diagram serupa dengan model data, namun kelas-kelas tidak hanya menunjuk struktur informasi tapi juga mendeskripsikan perilaku. Salah satu maksud class diagram adalah untuk mendefinisikan fondasi bagi diagram-diagram lain di mana aspek-aspek lain dari sistem ditunjukkan. Elemen-elemen yang penting dalam class diagram adalah sebagai berikut [18].

1. Kelas

Kelas merupakan elemen terpenting di sistem berorientasi objek. Kelas mendeskripsikan satu blok pembangunan sistem. Berikut adalah bagian dari kelas:

a. Nama
   kelas harus unik karena akan menjadi identifier di program.
b. Atribut
   Atribut adalah property bernama di kelas yang deskrpsikan range nilai yang dipunyai instan kelas. Kelas dapat mempunyai sejumlah atribut atau tidak sama sekali.
c. Operasi
   Operasi adalah implementasi layanan yang dapat pada sembarang objek kelas itu untuk mempengaruhi prilaku sistem

d. Acces Modifier

Acces Modifier dalam kelas digunakan untuk membatasi akses dari kelas lain yang ingin mengakses atribut atau operasi dari suatu kelas. Ada tiga *access modifier* yang digunakan yaitu *public* (+), *protected* (#) dan *private* (-)

2. Antarmuka

Antarmuka (Interface) merupakan korelasi operasi yang menspesifisikan layanan dari suatu kelas atau komponen. Antarmuka mendeskripsikan perilaku tampak secara eksternal dari elemen.

1. Kolaborasi

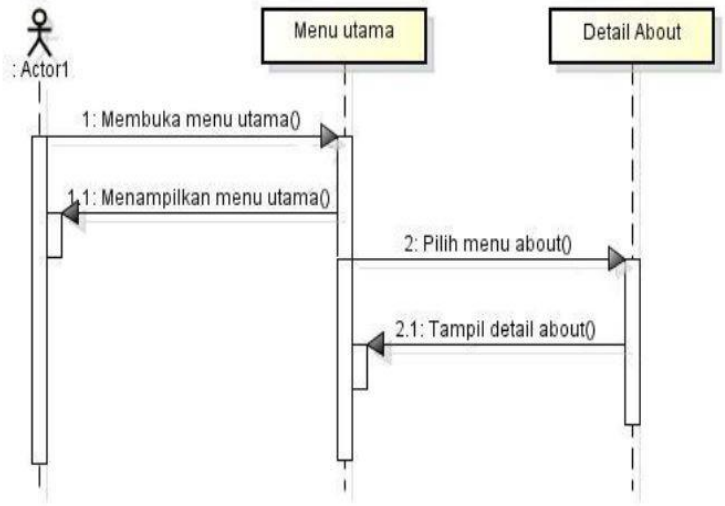
Kolaborasi merupakan pendefinisian suatu interaksi dan sekelompok peran elemen-elemen lain yang bekerja sama untuk menyediakan suatu perilaku kooperatif yang lebih besar dari penjumlahan seluruh elemen. Kolaborasi memiliki struktur, prilaku dan dimensi. Kolaborasi ini merepresentasikan implementasi pola tertentu yang membentuk sistem.

2. Hubungan

Hubungan antar kelas si diagram kelas terdiri dari Asosiasi dan Generalisasi.

2.16.4 Squence Diagram

Diagram seakan menggambarkan kelakuan/perilaku objek pada *use case,* dengan mengdeskripsikan waktu hidup objek dan message yang dikirimkan dan diteruma antar objek [18]. Objek-objek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut. Oleh karena itu untuk menggambarkan diagram skuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.



Gambar 2. 8 Contoh *Sequence Diagra*