BAB 2
TINJAUAN PUSTAKA

2.1 Citra

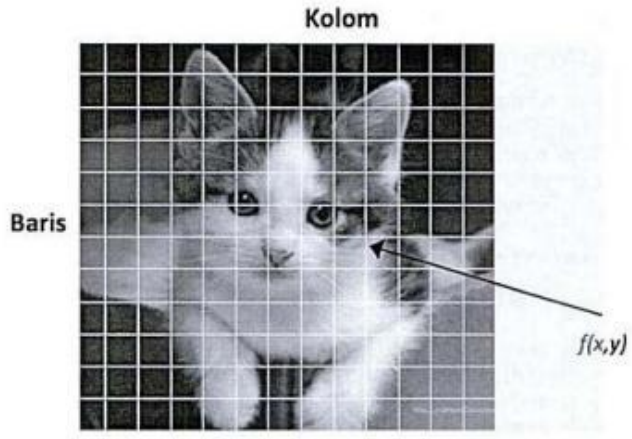
Citra adalah gambar dua dimensi yang dihasilkan dari gambar analog dua dimensi yang kontinu menjadi gambar diskrit melalui proses sampling[14]. citra dibedakan menjadi dua jenis yaitu citra analog dan citra digital. Citra merupakan keluaran dari suatu sistem perekam data yang bersifat optik, analog maupun digital. Perekaman data citra dibagi menjadi 2 yaitu Citra Analog dan Citra Digital.

2.1.1 Citra Analog

Citra analog adalah citra yang bersifat kontinu, 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 [14].

2.1.2 Citra Digital

Citra digital adalah gambar dua dimensi yang bisa ditampilkan pada layar komputer sebagai himpunan/diskrit nilai digital yang disebut pixel/picture elements. Dalam tinjauan matematis, citra merupakan fungsi kontinu dari intensitas cahata pada bidang dua dimensi[14]. Suatu citra dapat didefinisikan sebagai fungsi f(x,y) berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial, dan amplitudo f di titik koordinat (x,y) dianamakan intensitas atau tingkat keabuan dari citra pada titik tersebut. Apabila nilai x,y dan nilai amplitudo f secara keseluruhan berhingga (*finite*) dan bernilai diskrit maka dapat dikatakan bahwa citra tersebut adalah citra digital [15].



Gambar 2. 1l lustrasi Digitalitas Citra

2.2 Pengolahan Citra Digital

Pengolahan citra digital artinya melakukan pemrosesan gambar berdimenasi dua melalui komputer digital. Pengolahan citra adalah isitlah umum untuk berbagai teknik yang keberadaannya untuk memanipulasi dan memodifikasi citra dengan berbagai cara[15]. Foto adalah contoh gambar berdimensi dua yang dapat diolah dengan mudah. Setiap foto dalam bentuk citra digital dapat diolah menggunakan perangkat lunak tertentu. Misalnya citra yang gelap dapat diolah supaya menjadi lebih terang, merubah format warna menjadi grayscale, resize citra dan lain sebagainya.

2.3 *Pattern Recognition*

Pattern Recognition (PR) atau pengenalan Pola, merupakan salah satu cabang AI (*artifical intelegence)*[15], yang mempelajari pembuatan sebuah sistem untuk dapat mengenali suatu pola tertentu. Menurut Ursa Majorsy, Pola dalam hal ini merujuk pada pengertian suatu komposisi stimulus penginderaan yang kompleks yang dapat dikenali oleh manusia sebagai pengamat sebagai suatu kelompok objek. Rekognisi pola merupakan proses pengenalan kembali terhadap pola yang pernah dikenal. Oleh karena itu, jika kita melihat wajah teman kita atau mendengar lagu kesukaan kita, kita dapat mengenal masing-masing persepsi tersebut sebagai sesuatu yang sebelumnya telah dialami.

2.3.1 Penerapan *Pattern Recognition*

Seperti yang telah disebutkan diatas bahwa AI merupakan salah satu cabang Ilmu Komputer. Tapi karena kompleksitas area AI maka dibuatkan sub-sub bagian yang dapat berdiri sendiri dan dapat berdiri sendiri dan saling bekerja sama dengan sub bagian lain atau dengan disiplin ilmu lain[3]. 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.

2.4 Aksara Jepang

Pada umumnya tulisan Jepang dibagi menjadi empat jenis tulisan diantaranya adalah *hiragana, katakana, kanji dan romanji*. Perberdaan tulisan tersebut memiliki fungsi masing masing seperti digunakan dalam serapan bahasa

asing, nama orang, kata penghubung, dan kata baku [4]. Dalam tugas akhir ini peneliti mengambil studi kasus yaitu tulisan *katakana dan hiragana.*

1. *Katakana*

   Katakana untuk melambangkan suku kata tunggal, tetapi mempunyai fungsi yang berbeda dengan huruf *hiragana*. Huruf *katakana* selain digunakan untuk menulis kata-kata yang berasal dari bahasa asing, juga digunakan untuk penekanan suatu kata yang berasal dari Jepang asli. *Katakana* bentuk hurufnya terkesan kaku, karena setiap coretannya bersudut tajam sehingga katakana disebut sebagai huruf laki-laki. Huruf *katakana* berjumlah 46.

ア a	イ i	ウ u	エ e	オ o
カ ka	キ ki	ク ku	ケ ke	コ ko
サ sa	シ shi	ス su	セ se	ソ so
タ ta	チ chi	ツ tsu	テ te	ト to
ナ na	ニ ni	ヌ nu	ネ ne	ノ no
ハ ha	ヒ hi	フ fu	ヘ he	ホ ho
マ ma	ミ mi	ム mu	メ me	モ mo
ヤ ya		ユ yu		ヨ yo
ラ ra	リ ri	ル ru	レ re	ロ ro
ワ wa				ヲ (w)o
ン n				

Gambar 2. 2 Huruf Katakana

2. *Hiragana*

Pemakaian hiragana sangat luas pemakaiannya, terutama untuk menunjukan infleksi kata kerja dan kata-kata tertentu, yang akar katanya ditulis dengan karakter. Sama halnya dengan katakana, huruf hiragana mempunyai 46 huruf .

あ a	い i	う u	え e	お o
か ka	き ki	く ku	け ke	こ ko
さ sa	し shi	す su	せ se	そ so
た ta	ち chi	つ tsu	て te	と to
な na	に ni	ぬ nu	ね ne	の no
は ha	ひ hi	ふ fu	へ he	ほ ho
ま ma	み mi	む mu	め me	も mo
や ya		ゆ yu		よ yo
ら ra	り ri	る ru	れ re	ろ ro
わ wa				を wo
				ん n

Gambar 2. 3 Huruf Hiragana

2.4.1 Analisis Aksara Jepang

Huruf *Hiragana* adalah huruf asli jepang yang digunakan untuk menulis kata-kata asli bahasa jepang. Huruf hiragana terdiri atas 46 huruf pokok, yaitu 5 huruf vokal dan 41 huruf konsonan. Selain itu adalah huruf gabungan dan huruf konsonan ganda.

Huruf *Katakana* digunakan untuk penulisan kata-kata serapan yang diambil dari bahasa di luar bahasa jepang, nama-nama orang diluar nama orang jepang, dan nama-nama negara di luar Jepang, kecuali China dan Korea.

Huruf Hiragana dan Katakana mempunyai bentuk yang berbeda dan mempunyai pemakaian yang berbeda. Menurut Dedi Sutedi dalam bukunyadasar-dasar linguistik Jepang bahwa huruf hiragana dan katakana itu masing masing mempunyai 46 huruf, kedua huruf ini digunakan untuk melambangkan bunyi yang sama [4].

2.5 *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 [15]. Berikut adalah rumus konversi citra berwarna (RGB) menjadi grayscale.

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

Keterangan :

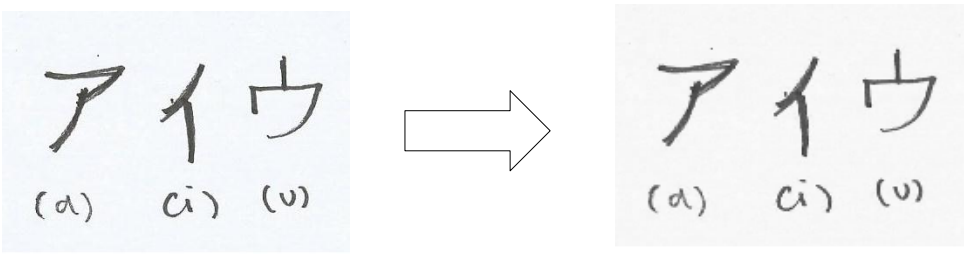
I = fungsi pencarian nilai skala keabu-abuan

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[16].



Gambar 2. 4 Contoh hasil *Grayscale*

2.5.2 *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* [17]. *Sauvola* termasuk dalam *locak threshold* dimana nilai ambang ditentukan oleh nilai pixel tetangga, tujuan

dilakukannya proses *threshold* adalah untuk menyederhanakan bentuk citra.

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

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

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

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 di bawah ini.

$$f(x,y) = \{^{0, img(x,y) < T(x,y)}_{255, img(x,y) \geq T(x,y)} \quad (2.5)$$

$$f(x,y) = \begin{cases} 1 ; x = 0 \\ 0 ; x = 255 \end{cases} \quad (2.6)$$

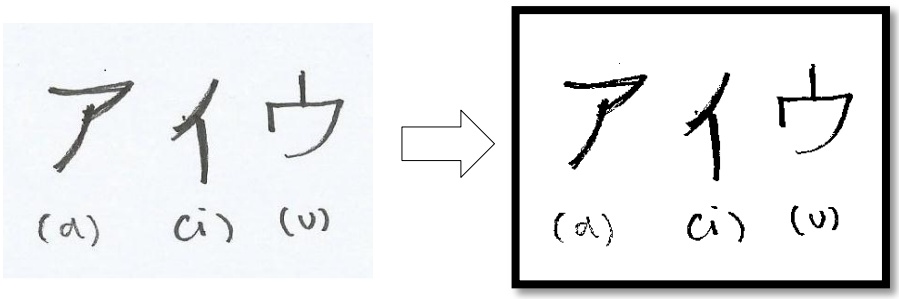
Keterangan :

f: fungsi yang menghasilkan nilai 0 atau 255

img: nilai grayscale citra

Nilai intensitas piksel yang lebih besar atau sama dengan nilai threshold akan dikonversi menjadi 0, maka dari itu nilai 255 berwarna

putih. Sedangkan nilai intensitas piksel yang kurang dari nilai threshold akan dikonversi menjadi 1, maka dari itu nilai 1 berwarna hitam.



Gambar 2. 5 Contoh hasil *Sauvola Threshold*

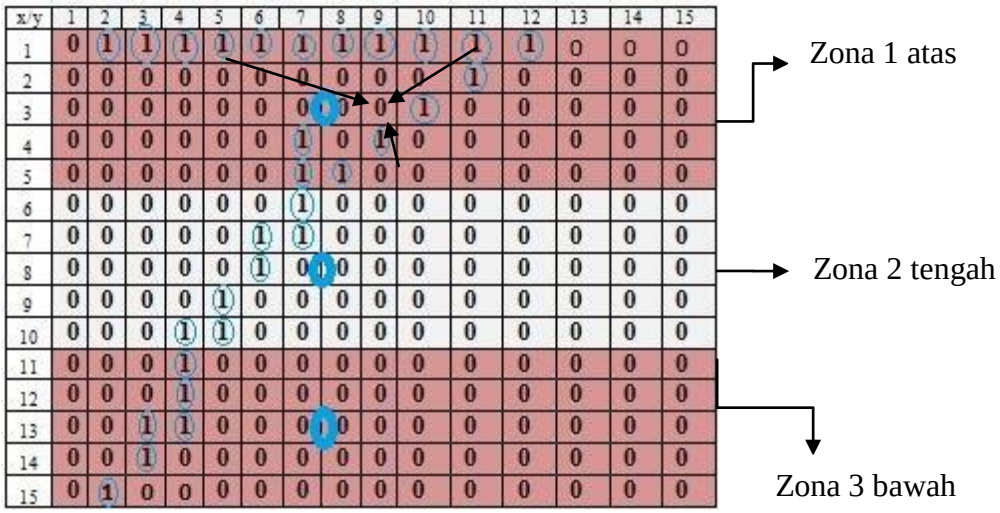
2.5.3 Ekstrasi Ciri

Ekstrasi fitur 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.[18].

Pada penelitian ini, penulis menggunakan metode ekstraksi *zoning*.

2.5.3.1 Ekstraksi Ciri Zoning

Zoning adalah salah satu ekstraksi fitur yang paling populer dan sederhana untuk diimplementasikan[18]. Sistem *optical recognition* (OCR) komersil yang dikembangkan oleh CELERA menggunakan metode zoning pada biner [18]. Setiap citra dibagi menjadi N x M zona dan dari setiap zona tersebut dihitung nilai fitur setiap zona adalah dengan menghitung jumlah pixel hitam setiap zona dan membaginya dengan jumlah pixel hitam terbanyak pada yang terdapat pada salah satu zona. Contoh pembagi 3 zona pada citra biner dapat dilihat pada gambar 2.6



Gambar 2. 6 Pembagian Zona pada Citra Biner

p_9	p_2	p_3
p_8	p_1	p_4
p_7	p_6	p_5

Gambar 2. 7 Pixel Ketetanggaan

Ada beberapa algoritma untuk metode ekstraksi ciri *zoning*, diantaranya metode ekstraksi ciri jarak metrik *Zone Centroid Zone*. Algoritma tersebut menggunakan citra digital sebagai input dan menghasilkan fitur untuk klasifikasi dan pengenalan sebagai *output*-nya. Berikut merupakan tahapan dalam proses ekstraksi ciri *Zone Centroid Zone* [9].

Dari gambar 2.6 hasil perhitungan jarak/ *Centroid* maka didapatkan sebagai berikut :

- Zona Atas : $x_c = 7{,}8\ y_c = 3$
- Zona Tengah : $x_c = 7{,}8\ y_c = 8$
- Zona Bawah : $x_c = 7{,}8\ y_c = 13$

Algoritma *Zone Centroid Zone* berdasarkan jarak metrik[8].

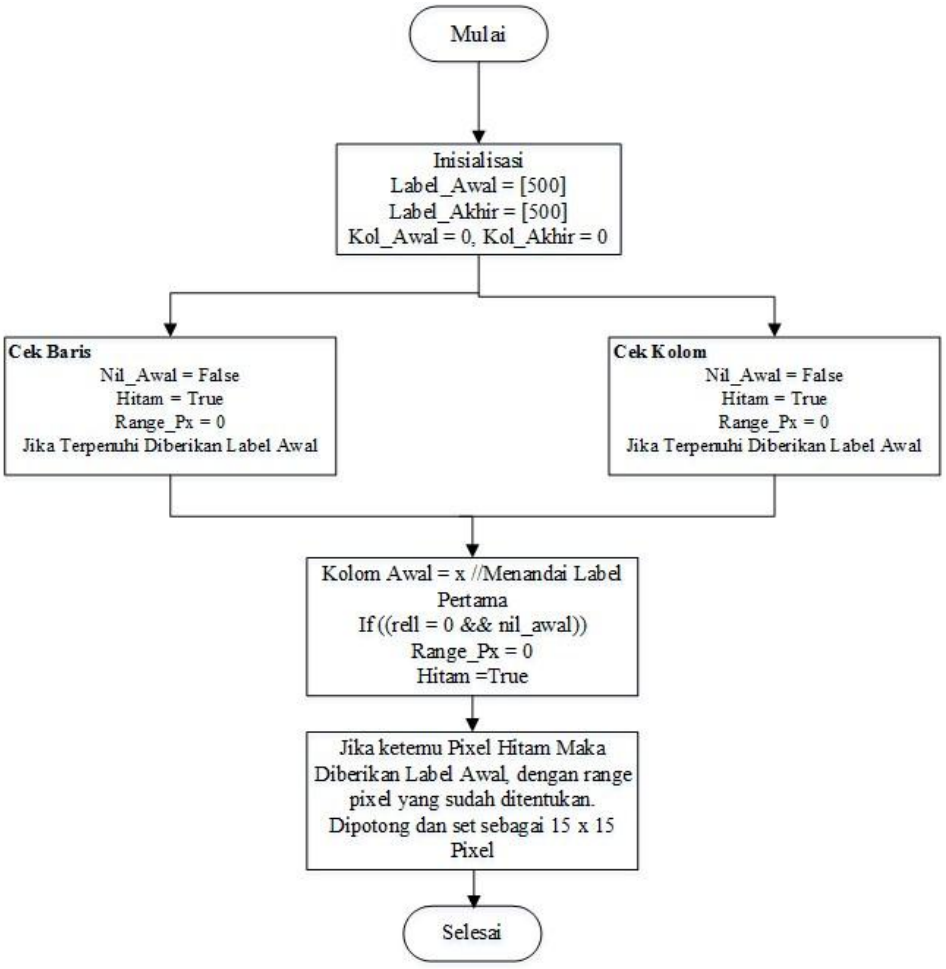
1. Bagi citra masukan ke dalam sejumlah *n* bagian yang sama.
2. Hitung *centroid* dari masing-masing zona.
3. Hitung jarak antara *centroid* masing-masing zona dan *pixel* yang ada di zona.
4. Ulangi langkah ke-3 untuk seluruh *pixel* yang ada di zona.
5. Hitung rata-rata jarak antara titik-titik tersebut.
6. Ulangi langkah ke 3-7 untuk setiap zona secara berurutan.
7. Hasilnya adalah *n* fitur yang akan digunakan dalam klasifikasi dan pengenalan.

2.6 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 [19]. Ada beberapa metode yang menggunakan teknik *learning* seperti *decision tree*, jaringan saraf tiruan, algoritma genetika dan lain-lain.

2.6.1 Segmentasi citra (*Image Segmentation*)

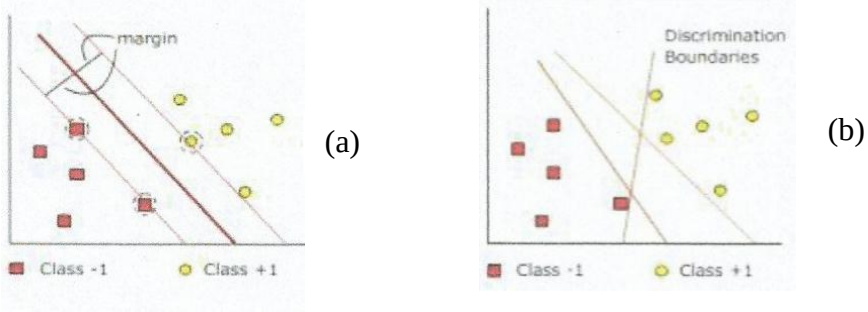
Segmentasi citra adalah teknik untuk membagi citra menjadi beberapa daerah atau region dimana setiap daerah memiliki kemiripan atribut[20]. Beberapa teknik segmentasi antara lain meliputi : pengembangan, penandaan, komponen terhubung, segmentasi berbasis cluster, dan transformasi hough, fokus pada segmentasi citra pada penelitian ini adalah menggunakan pengambangan dan penandaan komponen terhubung.



Gambar 2. 8 *Flowchart* segmentasi *Rule-Based*

2.6.2 *Support Vector Machine* (SVM)

Support Vector Machine (*SVM*) pertama kali diperkenalkan oleh Valdimir Vapnik pada tahun 1992 sebagai rangkaian harmonisasi konsep-konsep unggulan dalam bidang pengenalan pola [21]. *Support Vector Machine* merupakan metode pembelajaran yang digunakan untuk klasifikasi biner, ide dasarnya adalah mencari hyperlane terbaik yang berfungsi sebagai pemisah dua *class* pada *input space*[21].



Gambar 2. 9 *hyperlane* memisahkan dua kelas -1 dan +1

Berbeda dengan pendekatan neural network yang berusaha mencaru hyperlane pemisah antara class, SVM mencoba mencari dan memisahkan hyperlane yang terbaik yang berada pada dua kelas, gambar (a) menunjukan pola-pola yang merupakan anggota dari dua buah kelas +1 dan -1, pola yang berada pada anggota -1 kemudian disimbolkan dengan warna merah, sedangkan pola pada +1 warna kuning. Masalah pada klasifikasi dapat dilakukan dengan usaha menemukan garis (*hyperplane)* yang memisahkan kelompok tersebut. Pendefinisian persamaan suatu *hyperplane* pemisah yang dituliskan dengan :

$$w * x_i + b = 0 \qquad (2.6)$$

Data x_i yang terbagi kedalam dua kelas, yang termasuk kelas -1 (sampel negatif) didefinisikan sebagai vektor yang memenuhi pertidaksamaan (). Berikut.

$$w * x_i + b < 0 \; untuk \; y_i = -1 \qquad (2.7)$$

Sedangkang yang termasuk kelas +1 (sampel positif) memenuhi pertidaksamaan (2.8) berikut

$$w * x_i + b > 0 \; untuk \; y_i = +1 \qquad (2.8)$$

Dimana :

x_i = data input

y_i = label yang diberikan

w = nilai dari bidang normal

b = posisi bidang relatif terhadap koordinat

Parameter w dan b adalah parameter yang akan dicari nilainya, bila label y_i = -1, maka pembatas menjadi persamaan (2.9), berikut :

$$w * x_i + b \leq -1 \qquad (2.9)$$

Bila label data y_i = =1, maka pembatas menjadi persamaan (2.10), berikut :

$$w * x_i + b \geq +1 \qquad (2.10)$$

Margin terbesar dapat dicari dengan cara memaksimalkan jarak antara bidang pembatas kedua kelas dan titik terdekatnya, yaitu 2/|w|. Hal ini dirumuskan sebagai permasalahan *quadratic programming* (QP) *problem* yaitu mencari titik minimal persamaan (2.11) dengan memperhatikan persamaan (2.12)

$$\min \tau(w) = \frac{1}{2} ||w||^2 \qquad (2.11)$$

$$y_i(w * x_i + b) - \geq 0, (i = 1, \ldots, n) \quad (2.12)$$

Permasalahan ini dapat dipecahkan dengan berbagai teknik komputasi. Lebih mudah diselesaikan dengan mengubah persamaan (2.11) kedalam fungsi *lagrangian* pada persamaan (2.13) dan menyederhanakannya menjadi persamaan (2.14) berikut :

$$L(w, b, a) = \frac{1}{2} \left|\left|w\right|\right|^2 - \sum_{t=1}^{n} a_i \left(y_i \left((w^T x_i + b) - 1\right)\right) (2.13)$$

$$L(w, b, a) = \frac{1}{2} \left|\left|w\right|\right|^2 - \sum_{t=1}^{n} a_i \left(y_i \left((w^T x_i + b) + \sum_{i=1}^{n} a_i\right)\right) (2.14)$$

Dimana a_i adalah *larange multiplier* yang bernilai nol atau positif ($a_i \geq 0$). Nilai optimal dari persamaan (2.14) dapat dihitung dengan meminimalkan L terhadap *w*, *b*, dan *a*. Dapat dilihat persamaan (2.15) sampain (2.17) berikut :

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n} a_t y_i x_i = 0 \qquad (2.15)$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n} a_t y_i = 0 \qquad (2.16)$$

$$\frac{\partial L}{\partial a} = \sum_{i=1}^{n} a_t y_i (w^T x_i + b) - \sum_{i=1}^{n} a_i = 0 \qquad (2.17)$$

Maka masalah *lagrange* untuk klasifikasi dapat dinyatakan pada persamaan (2.18), berikut :

$$Min\ L(w, b, a) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} a_t y_i (w^T x_i + b) - \sum_{i=1}^{n} a_i \qquad (2.18)$$

Dengan memperhatikan persamaan (2.19) dan (2.20) berikut :

$$w - \sum_{i=1}^{n} a_i\, y_i x_i = 0 \qquad (2.19)$$

$$\sum_{i=1}^{n} a_i y_i = 0 \qquad (2.20)$$

Model persamaan (2.20) diatas merupakan model *primal lagrange*. Sedangkan denga maksimalkan L terhadap a_i, persamaannya menjadi persamaan (2.21) berikut :

$$max \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=1,j=1}^{n} a_i a_j y_i y_i \quad ^T x_i x_j^T \qquad (2.21)$$

Dengan memperhatikan persamaan (2.22), berikut :

$$\sum_{i=1}^{n} a_i y_i = 0\,, a_i \geq 0 (i, j = 1, \ldots, n) \qquad (2.22)$$

Untuk mencari nilai x_i dan y_i dapat dilakukan ketika sudah didapatkan nilai tiap karakter dari pembobotan td-idf dan inisialisasi kelas. Hasil dari pembobotan tf-idf diubah kedalam bentuk format data svm, sedangkan data kelas menjadi label.

Untuk mendapatkan nilai a_i, langkah pertama adalah mengubah setiap abstrak menjadi nilai vektor (*support vector)* = $(\frac{x}{y})$, kemudian nilai vektor dari setiap karakter dimasukkan ke persamaan (2.21) kernel *trick phi* berikut :

$$\emptyset[^{x}_{y}] = \{\sqrt{x_n^2 + y_n^2} > 2 \text{ maka } [\frac{\sqrt{x_n^2+y_n^2-x+|x-y|}}{\sqrt{x_n^2+y_n^2-x+|x-y|}}]$$

$$\sqrt{x_n^2} + y_n^2 \leq 2\, maka\, [^{x}_{y}] \qquad (2.23)$$

Nilai *x* didapatkan dari persamaan (2.24) kernel *linier* untuk *x* berikut :

$$\sum_{i=1,j=1}^{n} x_i x_j^T\, , (i, j = 1, \dots, n) \qquad (2.24)$$

Nilai *y* didapatkan dari persamaan (2.25) kernel *linear* untuk *y* berikut :

$$\sum_{i=1,j=1}^{n} y_i y_j^T\, , (i, j = 1, \dots, n) \qquad (2.25)$$

Untuk mendapatkan jarak tegak lurus yang optimal dengan mempertimbangkan vektor positif, maka hasil perhitungan dari subsitusi nilai *x* dan nilai *y* ke persamaan (2.25) diberi nilai bias = 1. Kemudian cari parameter a_i , dengan terlebih dahulu mencari nilai fungsi setiap abstrak menggunakan persamaan (2.26), lalu mencari nilaia_i, pada persamaan *liniear* menggunakan persamaan (2.27) dengan memperhatikan *i,j=1,...,n* berikut :

$$\sum_{i=1,j=1}^{n} a_i s_t^T s_j \qquad (2.26)$$

$$\sum_{i=1,j=1}^{n} a_i s_t^T s_j = y_i \qquad (2.27)$$

Setelah parameter a_i didapatkan kemudian masukkan ke permasamaan (2.28) berikut :

$$\overline{w} = \sum_{i=1}^{n} a_i s_i \qquad (2.28)$$

Hasil yang didapatkan menggunakan persamaan (2.28), selanjutnya digunakan persamaan (2.29), untuk mendapatkan nilai *w* dan *b* :

$$y = wx + b \qquad (2.29)$$

Sedemikian sehingga didapatkan nilai *w* dan *b* atau nilai *hyperlane* untuk mengklasifikasikan kedua kelas.

Sebuah fungsi bisa menjadi fungsi kernel jika memenuhi *Teorema mercer*,yang menyatakan bahwa matriks kernel yang dihasilkan harus bersifat *positive* semi *definite* [21]. Berikut ini adalah beberapa fungsi kernel yang umum digunakan yaitu :

a. Kernel *linier*

$$K(xi, x) = x^{iT}x \qquad (2.30)$$

b. *Polynomial*

$$K(xi, x) = (y.x^{iT}x + r)^{p}, y > 0 \qquad (2.31)$$

c. *Radial basis function*

$$K(xi, y) = \exp(-y|xi - x|^{2}), y > 0 \qquad (2.32)$$

d. *Sigmoid* kernel

$$K(xi, x) = \tanh(yxi^{T} + r) \qquad (2.33)$$

e. *Gaussian Radian basis function Kernel*

$$K(x' - x) = \exp(-||x' - x)^{2}//2\sigma 2 \qquad (2.34)$$

Hyperplane yang menjadi pemisah terbaik dapat ditemukan dengan mengukur margin *hyperlane* dan mencari titik maksimalnya. *Margin* adalah jarak antara hyperlane dengan *pattern* terdekat dari masing-masing *class.Pattern* yang paling dekat tersebut disebut *Support Vector*. Untuk pembahasan suatu kasus yang dapat dipisahkan secara linier, maka dalam hal ini fungsi pemisah yang dicari adalah fungsi linier. Fungsi tersebut dapat didefinisikan sebagai :

$$g(x) = sqn(f(x)) \qquad (2.35)$$

dengan

$$f(x) = WT\ X + b \qquad (2.36)$$

Hyperplane terbaik adalah hyperplane yang terletak ditengah-tengah antara dua objek dari dua kelas. Mencari *hyperlane* terbaik ekuivalen dengan memaksimalkan margin atau jarak dua set objek dari kelas yang berbeda.

Jika W$X_1 + b = +1$ adalah *hyperlane* pendukung dari kelas +1 (WX1 + b = +1) dan WX2 + b = -1 *hyperlane* pendukung dari kelas -1 (WX2 + b = -1), margin antara dua kelas dapat dihitung dengan mencari jarak antara kedua *hyperlane-hyperlane* pendukung dari kedua kelas. Secara spesifik *margin* dihitung dengan cara berikut :

$$(WX_1 + b = -1) - (WX_2 + b = -1) = W(x_1 - x_2 = 2) \quad (2.37)$$

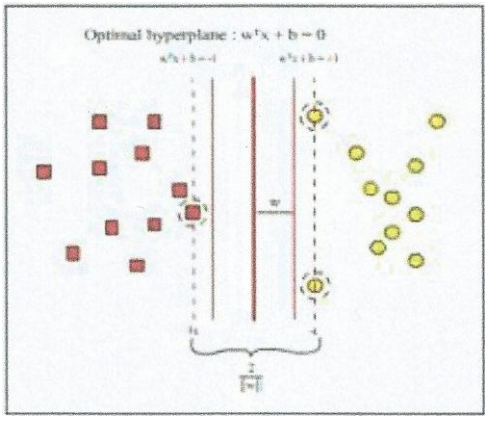
$$LD = \sum_{i=1}^{N} a_i - \frac{1}{2}\sum_{i,j} a_i a_j y_i y_j x_i x_j \quad (2.38)$$

Syarat 1 : $\sum_{i=1}^{N} a_i \, y_i = 0$

Syarat 2 : $a_i \geq 0{,}1 = 1{,}2, \ldots, N$

$x_i . x_j$ merupakan *dot = product* dua data dalam data latih. Hyperplane (batas keputusan atau pemisah).

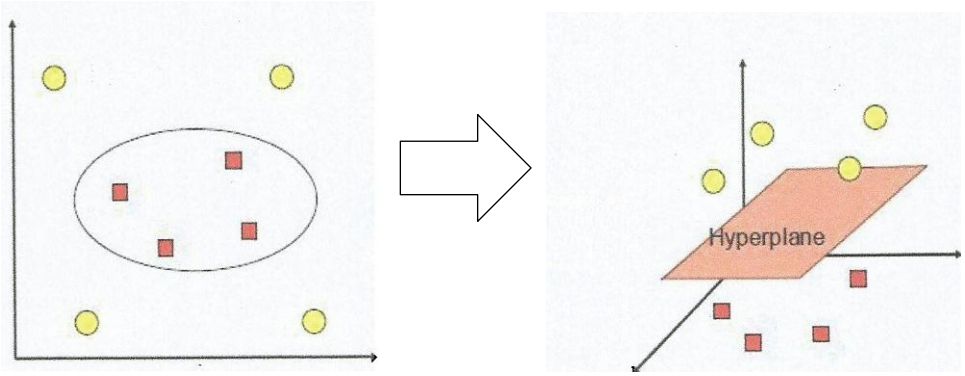
Gambar (2.8) menunjukan bagaimana SVM bekerja untuk menemukan suatu fungsi pemisah dengan margin yang maksimal[22].



Gambar 2. 10 fungsi pemisah optimal objek linier

Dalam feature space ini dot product menjadi $\varphi(x1), \varphi(x2)$ suatu fungsi kernel, $k(x, x')$, bisa untuk menggantikan dot product $(p(x), ip(x)'$. Kemudian feature space, kita bisa membuat suatu fungsi pemisah yang linear yang mewakili fungsi nonlinear di input space. Dalam input space, data tidak bisa dipisahkan secara linear, namun kita bisa memisahkan di

feature space. Karena itu dengan memetakan data ke feature space menjadi tugas klasifikasi menjadi lebih mudah[21].



Input Space X High-Dimensional Feature Space ϕ(X)

Gambar 2. 11 fungsi ruang vector kedua kelas linier oleh *hyperplane.*

Pemilihan kernel harus digunakan untuk sibtitusi dot product di feature space sangat bergantung pada data. Pemilihan fungsi kernel yang tepat adalah hal yang penting karena fungsi kerenel ini akan menentukan feature space dimana fungsi klasiner akan dicari. Sepanjang fungsi kernelnya legitimate, SVM akan beroperasi secara benar meskupin kita tidak tahu seperti apa map yang digunakan. Fungsi kernel yang legitimate diberikan oleh Teori Mercer [22]. Dimana fungsi tersebut harus memenuhi syarat : kontinus dan positive definite. Lebih mudah menemukan fungsi kernel daripada mencari map seperti apa yang tepat untuk melakukan mapping dari input space ke feature space.

2.7 *Support Vector Machine* untuk Multi-kelas

Pada awal perkembangan SVM pendekatan ini digunakan untuk klasifikasi dua kelas. Pengembangan ke arah persoalan klasifikasi untuk multi kelas masih menjadi perhatian para peneliti[8]. Terdapat dua pendekatan utama untuk SVM multi kelas, yang pertama kita dapat menemukan dan menggabungkan beberapa fungsi pemisah persoalan klasifikasi dua kelas untuk menyelesaikan persoalan klasifikasi dua kelas.

Kedua, secara langsung menggunakan semua data dari semua kelas dalam satu formulasi persoalan optimasi. Yang termasuk pada pendekatan pertama dimana beberapa fungsi untuk problem dua kelas dikembangkan lalu antara lain : satu-lawan-semua (*One-Againts-All, OAA)*, dan satu-lawan-satu (*One-Againts-One, OAO)*[8].

2.7.1 Metode Satu-Lawan-Semua (SLA)

Dengan menggunakan metode ini, dibangun k sebuah model SVM biner (k adalah jumlah kelas). Setiap mode klasifikasi ke-i dilatih dengan menggunakan keseluruhan data untuk solusi, terdapat permasalahan klasifikasi dengan 4 buah kelas. Untuk pelatihan digunakan 4 buah SVM biner seperti pada gambar 2.10 dan penggunaannya dalam mengklasifikasi data baru dapat dilihat pada gambar 2.10

$$min\frac{1}{2}(w^2)^T w^i + c\sum_{j=1}^{I}\varepsilon_j^i$$

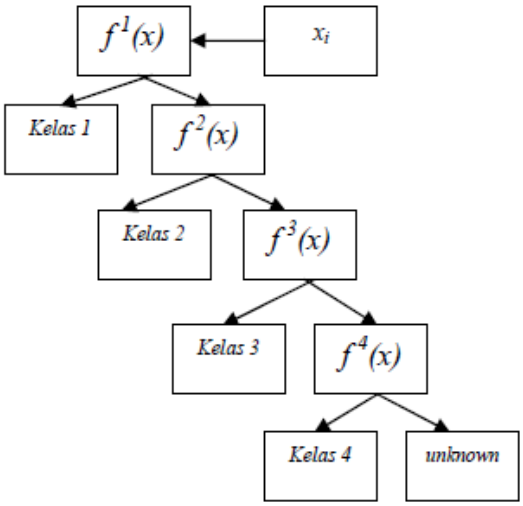
$$(w^i)^T\varphi(x_j) + b^i \geq -\varepsilon_j^i, \text{jika} y_j = i$$

$$(w^i)^T\varphi(x_j) + b^i \leq -\varepsilon_j^i, \text{jika} y_j \neq i$$

$$\varepsilon_j^i \geq 0, j = 1, \dots, l. \qquad (2.39)$$

Tabel 2. 1 Contoh 4 SVM biner dengan metode One-Against-All

$y_i = 1$	$y_i = -1$	Hipotesis
Kelas 1	Bukan kelas 1	$f^1(x) = (w^1)x + b^1$
Kelas 2	Bukan kelas 2	$f^2(x) = (w^2)x + b^2$
Kelas 3	Bukan kelas 3	$f^3(x) = (w^3)x + b^3$
Kelas 4	Bukan kelas 4	$f^4(x) = (w^4)x + b^4$



Gambar 2. 12 Contoh klasifikasi *One Against All*

Pada gambar 2.12 diatas diagram kelas dalam penelitian ini akan menjadi kelas huruf aksara katakana dan hitagana yang berjumlah 96 kelas.

2.8 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[24].

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[24], 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.*

2.8.1 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*[24].

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.

2.8.2 Pengujian Akurasi

Akurasi merupakan seberapa dekat suatu angka hasil pengukuran terhadap angka sebenarnya (*true value* dan *reference value*). Tingkat akurasi diperoleh dengan persamaan sebagai berikut.

$$Akurasi = \frac{jumlah\ karakter\ sama}{jumlah\ seluruh\ karakter}\ X\ 100\% \qquad (2.40)$$

2.9 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.*

2.9.1 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 [26].

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*) [26]. 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[20], 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.

2.10 UML (*Unified Modeling Language*)

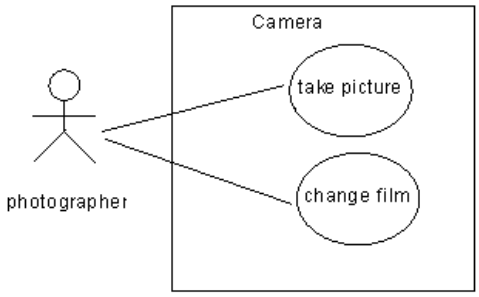
Unified Modeling Language (UML) adalah notasi yang lengkap untuk membuat visualisasi model suatu sistem. Sistem informasi dan fungsi, tetapi secara normal digunakan untuk memodelkan sistem komputer. Didalam pemodelan objek guna menyajikan sistem yang berorientasi pada objek dan pada orang lain, akan sangat sulit dilakukan jika pemodelan tersebut dilakukan dalam bentuk kode bahasa pemrograman [27].

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. Alat-alat dari sistem perangkat lunak. UML biasanya disajikan dalam bentuk diagram atau gambar yang meliputi *class* beserta atribut dan operasinya, serta hubungan antara kelas. UML terdiri dari banyak diagram diantaranya *use case diagram , activity diagram, class diagram,* dan *squence diagram*.

2.10.1 *Use Case Diagram*

Dalam konteks UML, tahap konseptualisasi dilakukan dengan pembuatan *use casediagram* yang sesungguhnya merupkan deskripsi peringkat tinggi bagaimana perangkat lunak (aplikasi) akan digunakan oleh penggunanya.

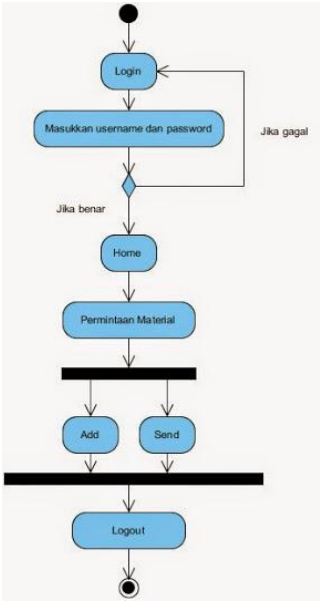
Use Case Diagram merupakan deskripsi lengkap tentang interaksi yang terjadi antara para akor dengan sistem [27]. Dalam hal ini, setiap objek yang berinteraksi dengan sistem merupakan aktor untuk sistem berprilaku kepada aktronya. Aktor dalam *use case diagram* digambarkan sebagai ikon yang berbentuk manusia, sementara *use case* 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 dituliskan dengan kata kerja.



Gambar 2. 13 Contoh Penggunaan *Use Case Diagram*

2.10.2 Activity Diagram

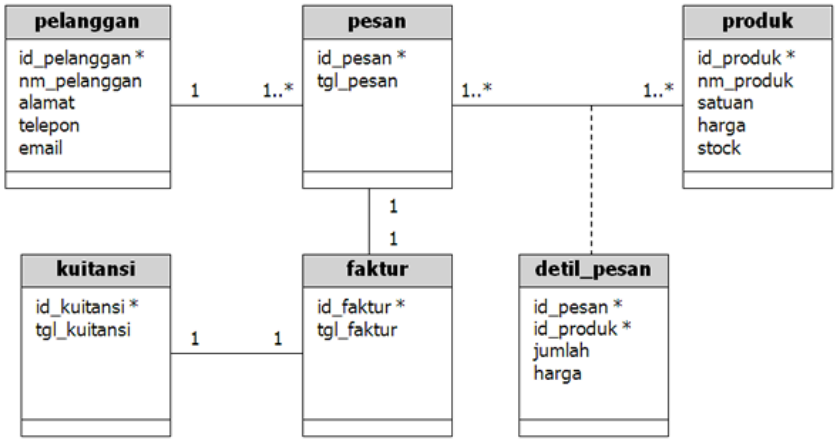
Activity Diagram pada dasarnya menggambarkan skenario secara grafis. *Activity Diagram* cukup serupa dengan diagram alur (*flow chart*), yang membedakan hanya *swimlane* yang menunjukkan suatu *state* berada pada objek/kelas tertentu. Keunggulan dari *activity diagram* adalah lebih mudah dipahami dibanding skenario. Selain itu, dengan menggunakan *activity diagram,* kita dapat melihat dibagian manakah sistem dari suatu skenario akan berjalan [27].



Gambar 2. 14 Contoh *Activity Diagram*

2.10.3 *Class Diagram*

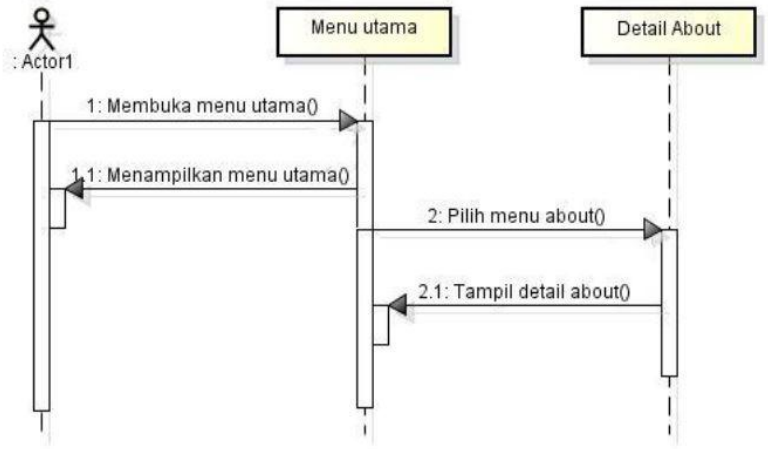
Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. *Class Diagram* menggambarkan struktur dan deskripsi kelas, package, dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi dan sebagainya.



Gambar 2. 15Contoh *Class Diagram*

2.10.4 *Sequence Diagram*

Diagram seakan menggambarkan kelakuan/perilaku objek pada *use case*, dengan mengdeskripsikan waktu hidup objek dan message yang dikirimkan dan diteruma antar objek [27]. 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. 16 Contoh *Sequence Diagram*