BAB 2
TINJAUAN PUSTAKA

2.1 Pengolahan Citra

Istilah citra digital sangat populer pada masa sekarang. Banyak perlatan elektronik,misalnya *scanner*, kamera digital, mikroskop digital, dan *fingerprint reader* (pembaca sidik jari), yang menghasilkan citra digital. Perangkat lunak untuk mengolah citra digital jugasangat popular digunakan oleh pengguna untuk mengolah foto atau untuk berbagai keperluan lain. Penolahan citra merupakan proses memanipulasi dan menganalisis citra dengan bantuan komputer, dalam hal ini mengolah informasi yang terdapat pada suatu gambar untuk keperluan pengenalan objek secara otomatis. Ada berbagai teknik pengolahan citra tergantung kebutuhan dan keluaran yang diinginkan [10].

2.2 Karya Tulis Ilmiah Skripsi

Salah satu jenis karya tulis tulis ilmiah yaitu skripsi. Skripsi adalah istilah yang digunakan di Indonesia untuk mengilustrasikan suatu karya tulis ilmiah berupa paparan tulisan hasil penelitian sarjana S1 yang membahas suatu permasalahan/ fenomena dalam bidang ilmu tertentu dengan menggunakan kaidah-kaidah yang berlaku. Skripsi ditulis berdasarkan kajian pada studi pustaka, penyelidikan, observasi, atau penelitian lapangan. Penelitian yang akan dibuat ini menggunakan hasil *scan* beberapa bagian dari karya tulis ilmiah skripsi dalam bentuk gambar/citra untuk dikenali setiap karakter yang terdapat didalamnya.

2.3 *Pre-Processing*

Pre-processing adalah bagisan penting dari setiap sistem pemrosesan Bahasa alami, karena karakter, kata, dan kalimat yang diintifikasi pada tahap ini adalah unit dasar atau awal sebelum pemrosesan lebih lanjut. *Pre-processing* dilakukan karena data teks sering mengandung berbagai macam format atau berbeda-beda seperti format angka dan kata-kata yag tidak membantu dan dapat dihilagkan sehingga memudahkan proses selanjutya.

Image preprocessing adalah suatu bentuk pengolahan atau pemrosesan sinal dengan input berupa gambar (*image*) dan ditransformasikan menjadi gambar lain

sebagai keluatan dengan teknik tertentu. *Image preprocessing* dilakukan untuk memperbaiki kesalahan data sinyal akibat transmisi dan selama akusisi sinyal, serta untuk meningkatkan kualitas penampakan gambar agar lebih mudah diinterpretasi oleh sistem penglihatan manusia baik dengan melakukan manipulasi dan juga penganalisisan terhadap gambar. Operasi *image preprocessing* dapat dikelompokan berdasarkan tujuan transformasinya.

2.3.1 *Grayscaling*

Grayscale adalah istilah untuk menyebutkan satu citra yang memiliki warna abu-abu, hitam, dan putih. *Grayscale* adalah koleksi atau kisaran corak monokromik (abu-abu), mulai dari putih murni di ujung yang paling terang hingga hitam murni di ujung yang berlawanan.

Citra *grayscale* adalah citra yang hanya memiliki 1 buah kanal sehinga yang ditapilan hanyalah nilai intensitas atau dikenal juga dengan istilah derajat keabuan. Karena jenis citra ini hanya memiliki 1 kanal saja, maka citra *grayscale* memiliki tempat penyimpanan yang lebih hemat. Jenis citra ini disebut juga sebagai 8-bit. Foto hitam putih maupun gambar yang ditampilkan oleh televisi hitam putih sebenarnya meggunakan citra *grayscale,* bukan dalam warna hitam dan warna putih. Namun dikalagan masyarakat istilah foto hitam putih maupun televisi hitam putih sudah terbiasa digunakan falam kehidupan sehari-hari [12].

Secara teori ada beberapa cara dalam mengonversi citra berwarna RGB ke dalam citra *grayscale*. Untuk mendapatkan hasil honversi yang lebih baik, persamaan berikut dapat digunakan

$$y' = 0.299R' + 0.587G' + 0.144B' \quad (2.1)$$

Keterangan :

y’ : Citra *grayscale*

R’ : Komponen merah dari citra RGB

G’ : Komponen hijau dari citra RGB

B’ : Komponen biru dari citra RGB

2.3.2 *Thresholding*

Thresholding yaitu mengubah citra grayscale menjadi citra berwarna hitam putih dengan nilai ambang yang sudah ditentukan. Terdapat dua metode thresholding, yaitu global thresholding dan local thresholding [27]. Apabila nilai ambang (*threshold*) bergantung hanya pada satu nilai aras keabuan, pengambangan disebut global, dimana semua piksel dalam citra akan ditentukan oleh satu nilai ambang yang sudah ditentukan. Pengambangan ini biasa digunakan untuk memisahkan tulisan hitam yang berada diatas secarik kertas putih. Namun, jika nilai ambang berg maka disebut pengambangan lokal. Dimana nilai ambang untuk setiap piksel ditentukan oleh nilai piksel tetangga, maka nilai ambang untuk masing-masing piksel belum tentu sama. Perbedaan ciri kedua jenis thresholding ini yaitu sebagai berikut.

a. Ciri-ciri global thresholding

Ciri-ciri dari global thresholding yaitu [27]:

1) Tidak memperhatikan hubungan spasial antarpiksel.
2) Sensitif terhadap pencahayaan tidak seragam.
3) Hanya berlaku untuk keadaan ideal (misalnya, latar belakang berwarna putih dan objek berwarna hitam).
4) Bergantung kepada pemilihan nilai ambang yang tepat.

b. Ciri-ciri local thresholding

Ciri-ciri dari local thresholding yaitu [27]:

1) Memperhatikan hubungan spasial antarpiksel.
2) Mampu beradaptasi dengan pencahayaan yang tidak seragam.
3) Berlaku untuk keadaan ideal atau tidak.antung pada beberapa nilai sesuai dengan aras keabuan pada citra

2.3.3 Binerisasi

Dalam pengolahan citra digital, proses binerisasi adalah mengubah citra grayscale menjadi citra biner, artinya mengubah warna tiap-tiap piksel pada citra bernilai 0 dan 255 ke dalam piksel bernilai 0 dan 1. Sehingga citra hanya berwarna hitam dan putih. Pada proses binerisasi, menggunakan nilai ambang (*threshold*)

untuk menentukan nilai *grayscale* tertentu yang diubah menjadi piksel bernilai 0 atau 1. Pada penelitian ini, digunakan nilai tengah *threshold* yang diambil dari pembagian total intensitas yaitu 256 dibagi 2 dimana hasilnya 128. Jika nilai *grayscale* di atas 128 maka nilai tersebut diubah menjadi nilai 0 dan jika nilai *grayscale* di bawah 128 maka nilai tersebut di ubah menjadi 1 [12]. Berikut adalah persamaan yang digunakan.

$$g(x,y) = \begin{cases} 0 \; if \; f(x,y) \geq 128 \\ 1 \; if \; f(x,y) < 128 \end{cases} \quad (2.2)$$

Keterangan :

$g(x,y)$: Nilai biner

$f(x,y)$: Nilai citra *grayscale* suatu piksel

0 : Hitam

1 : Putih

2.4 Segmentasi

Segmentasi bertujuan untuk memotong huruf per-huruf. Pemotongan tersebut dilakukan dengan cara mencari pixel-pixel terluar dari setiap sisi (atas, bawah, kiri, kanan). Pixel-pixel terluar itulah yang akan menjadi batas pemotongan, sehingga didapat citra segiempat yang siap diproses lebih lanjut.

2.4.1 *Profile Projection*

Metode *Profile Projection* digunakan untuk memisahkan tulisan perbaris dan perkarakter. Metode *Profile Projection* akan menghitung jumlah piksel *non-background* secara vertikal dan horizontal dan nilai tersebut akan dibandingkan dengan nilai ambang tertentu untuk memisahkan tulisan perbaris dan perkarakter.

1. *Profile Projection* secara horizontal

   Profile Projection secara horizontal adalah jumlah banyaknya piksel hitam yang tegak lurus dengan sumbu x. Profile Projection secara horizontal dipresentasikan dengan suatu vektor Ph berukuran M. Profile Projection secara horizontal pada kolom ke-j, yaitu Ph[j], didefinisikan sebagai berikut :

$$P_h[h] = \sum_{j=1}^{N} S[i,j] \quad (2.3)$$

2. *Vertical Profile Projection* secara vertical

Profile Projection secara vertikal adalah jumlah banyaknya piksel hitam yang tegak lurus dengan sumbu y. *Profile Projection* secara vertical dipresentasikan dengan suatu vektor Pv berukuran N. *Profile Projection* secara vertikal pada baris ke-I, yaitu Pv[i], didefinisikan sebagai berikut :

$$P_v[v] = \sum_{i=1}^{M} S[i,j] \quad (2.4)$$

Keterangan :

S : Citra Biner

M : Banyak kolom pada citra

N : Banyak baris pada citra

2.4.2 *Line Segmentation*

Tahap selanjutnya adalah melakukan proses *Line Segmentation* yang digunakan untuk memisahkan tulisan tiap baris. Metode yang digunakan pada *Line Segmentation* yang berkerja dengan *Horizontal Profile Projection* yang berkerja dengan menghitung jumlah piksel berwarn hitam pada citra secara horizontal (sumbu x) [14]. Pemotongan citra perbaris dilakukan dengan melihat *gap* kosong yang ada pada setiap baris.

2.4.3 *Character Segmentation*

Character segmentation dilakukan untuk memusahkan tiap karakter pada citra hasil segmentasi kata. Segmentasi yang dilakukan untuk tiap karakter dilakukan dengan menggunakan metode yang sama dengn segmentasi kata yaitu metode *Vertical Profile Projection* yang menghitung jumlah piksel hitam pada sumbu y (*vertical*). Pemisahan untuk tiap karakter, dilakukan dengan memanfaatkan gap yang ada pada tiap kata.

2.5 *Resize* (normalisasi ukuran citra)

Resizing adalah proses yang digunakan untuk mengubah ukuran citra digital dalam pixel, baik menjadi lebih kecil atau lebih besar dari ukuran sebenarnya. Salah satu metode yang sering digunakan dalam pengubahan ukuran citra adalah metode *Nearest Neighbor*. *Nearest Neighbor* merupakan metode interpolasi paling sederhana dan cepat dengan memindahkan ruang yang kosong dengan piksel yang

berdekatan (the *nearest neighboring pixel*) pada saat pengecilan atau pembesaran skala gambar [26].

Nearest neighbor menggunakan nilai piksel terdekat pada gambar awal untuk memberikan nilai piksel pada gambar awal yang akan diperbesar atau diperkecil. Sebagai contoh terdapat sebuah gambar dengan ukuran 4 x 4 dengan jumlah piksel 16 dimana setiap pikselnya diwakilkan dengan nilai A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P. Kemudian gambar akan diperbesar menjadi ukuran 6 x 8 dengan jumlah piksel 48 menggunakan *Nearest neighbor* [27]. Berikut adalah contoh dari *nearest neighbor*.

Tabel 2.1 Nilai Piksel Citra Asli

(x,y)	1	2	3	4
1	A	B	C	D
2	E	F	G	H
3	I	J	K	L
4	M	N	O	P

Adapun contoh proses perhitungan untuk mendapatkan setiap nilai piksel pada gambar dengan ukuran 6 x 8 yaitu:

Perbandingan lebar (*ratio weight)* = 4 : 6 = 2 : 3.

Perbandingan panjang (*ratio height)* = 4 : 8 = 1 : 2.

1) Posisi piksel dengan nilai x = 1, y = 1

   $Piksel_x$ = *ceil*(x * *ratio weight*) = *ceil*(1 * 2/3) = 1

   $Piksel_y$ = *ceil*(y * *ratio height*) = *ceil*(1 * ½) = 1

   Nilai piksel pada gambar ukuran 6 x 8 dengan x = 1 yang menghasilkan $Piksel_x$ = 1 dan y = 1 yang menghasilkan $Piksel_y$ = 1 disesuaikan dengan nilai piksel pada gambar awal dengan i = 1 dan j = 1 yaitu A.

2) Posisi piksel dengan nilai x = 1, y = 2

   $Piksel_x$ = *ceil*(x * *ratio weight*) = *ceil*(1 * 2/3) = 1

   $Piksel_y$ = *ceil*(y * *ratio height*) = *ceil*(2 * ½) = 1

Nilai piksel pada gambar ukuran 6 x 8 dengan x = 1 dan y = 2 juga memiliki nilai A.

Ceil (*ceiling*) merupakan proses pembulatan sebuah bilangan ke atas. Sehingga akan menghasilkan citra matriks berikut.

Tabel 2.2 Nilai Piksel Hasil *Resize*

(x,y)	1	2	3	4	5	6
1	A	B	B	C	D	D
2	A	B	B	C	D	D
3	E	F	F	G	H	H
4	E	F	F	G	H	H
5	I	J	J	K	L	L
6	I	J	J	K	L	L
7	M	N	N	O	P	P
8	M	N	N	O	P	P

Pada penelitian ini *resize* digunakan untuk merubah ukuran *pixel* sesuai dengan kebutuhan dalam melakukan proses ektraksi ciri. Hal ini dilakukan agar semua citra karakter dari hasil segmentasi mempunyai ukuran yang sama (normalisasi ukuran citra) sebelum masuk ke tahap ekstraksi fitur.

2.6 Ektraksi Fitur Zoning

Ekstraksi Ciri *Zoning* merupakan salah satu metode ekstaksi ciri dimana inti dari metode ini adalah citra dibagi pada sejumlah zona yang sama untuk dikenali ciri dari setiap karakter huruf yang selanjutnya menghasilkan sebuah nilai untuk diproses pada tahapan klasifikasi [25].

Ada beberapa algoritma untuk metode ekstraksi ciri *zoning*, diantaranya metode ekstraksi ciri jarak metrik ICZ *(image centroid zone),* metode ekstraksi ciri jarak metrik ZCZ *(Zone centroid and zone).* Kedua algoritma tersebut menggunakan citra digital sebagai input dan menghasilkan fitur untuk klasifikasi dan pengenalan sebagai *output*-nya. Berikut merupakan tahapan dalam proses ekstraksi ciri ICZ.

1. Hitung centroid dari citra masukan menggunakan persamaan berikut.

$$x_c = \frac{(i_1.p_1 + i_2.p_2 + \cdots + i_n.p_n)}{(p_1 + p_2 + \cdots + p_n)} \quad (2.5)$$

$$y_c = \frac{(j_1.p_1 + j_2.p_2 + \cdots + j_n.p_n)}{(p_1 + p_2 + \cdots + p_n)} \quad (2.6)$$

Keterangan:

x_c = *centroid* koordinat *x*

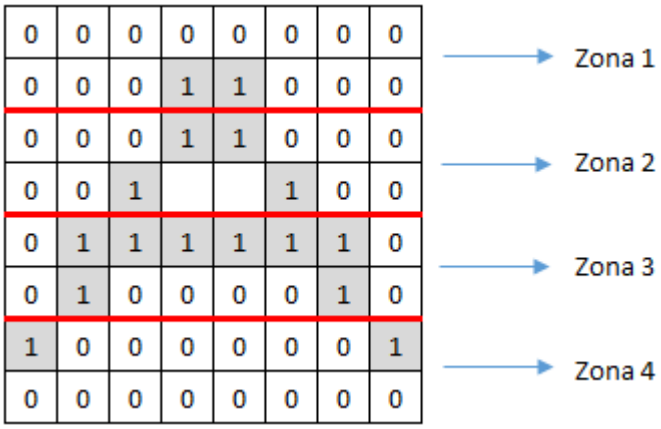
y_c = *centroid* koordinat *y*

x_n = koordinat *x* dari piksel ke-*n*

y_n = koordinat *y* dari piksel ke-*n*

p_n = nilai piksel ke-*n*

2. Bagi citra masukan ke dalam *n* zona yang sama. Contoh pembagian 4 Zona pada citra biner dapat dilihat pada gambar 2.3.



Gambar 2.1 Pembagian Zona Citra Biner

3. Hitung jarak antara centroid citra dengan masing-masing piksel yang ada dalam zona.

$$jarak = \sqrt{(x_p - x_c)^2 + (y_p - y_c)^2} \quad (2.7)$$

Keterangan:

jarak = jarak antara koordinat centroid *(x,y)* dengan koordinat objek

x_p = koordinat *x* objek

y_p = koordinat y objek

x_c = *centroid* koordinat *x*

y_c = *centroid* koordinat *y*

4. Ulangi langkah ke 3 untuk setiap piksel yang ada di zona

5. Hitung rata-rata jarak antara titik-titik tersebut.
6. Ulangi langkah-langkah tersebut untuk keseluruhan zona,
7. Hasilnya adalah *n* fitur yang akan digunakan dalam klasifikasi dan pengenalan.

2.7 *Smooth Support Vector Machine* (SSVM)

SSVM adalah pengembangan SVM dengan menggunakan teknik *smoothing*. Metode ini pertama kali diperkenalkan oleh Lee pada tahun 2001 [6]. SVM memanfaatkan optimasi dengan quadratic programming, sehingga untuk data berdimensi tinggi dan data jumlah besar SVM menjadi kurang efisien. Oleh karena itu dikembangkan smoothing technique yang menggantikan plus function SVM dengan integral dari fungsi sigmoid neural network yang selanjutnya dikenal dengan *Smooth Support Vector Machine* (SSVM).

Keterangan :

w^b : Seukuran Inputan

b^0 : Angka

D : Matrik Diagonal berukuran (m x m) dengan nilai [1, -1]

A : Matrik berukuran (m x n)

m : Banyaknya data

n : fitur

w : vektor normal berukuran (n x 1)

e : vektor berukuran (m x 1)

γ : Parameter penentu lokasi bidang pemisah terhadap titik asal

v : Parameter positif yang menyeimbangkan bobot dari training error dan margin maximation term

Diberikan masalah klasifikasi dari *n* objek dalam ruang dimensi R^p sehingga susunan data berupa matriks *A* berukuran *n x p* dan keanggotaan tiap titik terhadap kelas {+1} atau {-1} yang didefinisikan pada diagonal matriks D berukuran ,problem optimasinya adalah

$$min_{w,b,\xi} \quad \frac{c}{2}\xi'\xi + \frac{1}{2}\,(w'w + b^2) \tag{2.8}$$

dengan kendala

$$D\ (Aw + eb) + \ \xi\ \ \geq e, \xi\ \geq 0 \tag{2.9}$$

Solusi problem adalah

$$\xi\ = \big(e - D(Aw + eb)\big) \tag{2.10}$$

Dimana ξ adalah variabel *slack* yang mengukur kesalahan klasifikasi. Kemudian dilakukan substitusi dan konversi, sehingga persamaan dapat ditulis sebagai berikut:

$$min_{w,b}\ \ \frac{c}{2}||(e - D(Aw - eb)||_2^2 + \frac{1}{2}\ (w'w + b^2) \tag{2.11}$$

Fungsi objektif dalam persamaan tidak memiliki turunan kedua. Teknik *smoothing* yang diusulkan dilakukan dengan mengganti fungsi plus dengan p(x,a) yaitu integral dari fungsi sigmoid *neural network* atau dapat dituliskan sebagai berikut:

$$p\ (x, a) = x + \frac{1}{a}\log(1 + \ \varepsilon^{-ax}), a > 0 \tag{2.12}$$

dimana α adalah parameter *smoothing*. Dengan menggantikan fungsi plus dengan p (x,a) maka diperoleh model SSVM sebagai berikut:

$$\mathbf{min}_{(\mathbf{w},\mathbf{b})\in\mathbf{Rp+1}}{}^{\phi a}(\mathbf{w}, \mathbf{b}) \tag{2.13}$$

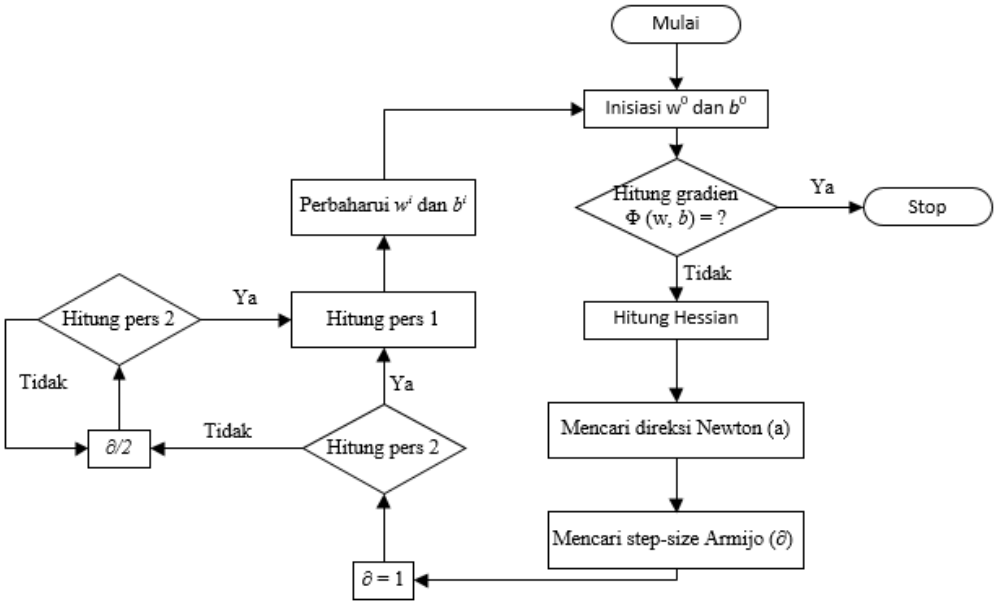
$$= \mathbf{min}_{(\mathbf{w},\mathbf{b})\in\mathbf{Rp+1}}\ \ \frac{\mathbf{c}}{\mathbf{2}}||\mathbf{p}(\mathbf{e} - \mathbf{D}(\mathbf{Aw} - \mathbf{eb})||_2^2 + \frac{\mathbf{1}}{\mathbf{2}}\ (\mathbf{w}'\mathbf{w} + \mathbf{b}^2)$$

Secara umum, problem optimasi SSVM dapat ditulis sebagai berikut:

$$min_{(w,b)\in Rp+1}{}^{\phi a}(w, b) = min_{(w,b)\in Rp+1}\ \ \frac{c}{2}||p(e - \tag{2.14}$$

$$D\big(K\big(x_i, x_j\big)Dw - eb\big), a)||_2^2 + \frac{1}{2}\ (w'w + b^2)$$

Yang diselesaikan dengan iterasi Newton Armijo (Gambar 2.5) dan *K(Xi,Xj)* merupakan fungsi kernel yang dalam penelitian ini digunakan kernel Gaussian atau bisa dirumuskan berikut *K(Xi,Xj) = exp (-y(*||Xi – Xj||^2) dengan parameter kernel *y*.



Gambar 2.2 Diagram Alir Algoritma Newton-Armijo

Persamaan 1 :

$$\phi_a(w_i, b_i) - \phi_a((w_i, b_i) + (\lambda_i d_i) \geq -\delta\lambda_i \nabla\phi_a(w_i, b_i)d_i \quad (2.15)$$

Persamaan 2 :

$$w_{i+1,} b_{i+1} = (w_i, b_i) + (\lambda_i d_i) \quad (2.16)$$

Saat iterasi pada algoritma Newton-Armijo berhenti, diperoleh nilai *w* dan *b* yang konvergen. Dengan demikian fungsi pemisah yang diperoleh untuk kasus klasifikasi linier adalah

$$F(x) = sign\ (w'x + b) \quad (2.17)$$

Sedangkan fungsi permisah untuk kasus klasifikasi nonlinier adalah sebagai berikut

$$F(x) = sign\ (w'x + b) = sign\ (u'D'K(Xi,Xj) + b) \quad (2.18)$$

Perumusan program linier SVM 1-norm adalah salah satu cara untuk memilih atribut (*feature selection*) diantara varian-varian norm SVM, problem linier tersebut adalah sebagai berikut:

$$\min_{(w,b,s,\xi)\in R^{(2p)+1+n}} Ce'\xi + e's \tag{2.19}$$

Dengan kendala

$$D(Aw + eb) + \xi \geq e \tag{2.20}$$

$$-s \leq w \leq s$$

$$\xi \geq 0 \tag{2.21}$$

Solusi dari ***w*** mampu menghasilkan model yang parsimoni dan bersifat *sparsity*. Jika nilai dari elemen vektor ***wp*** = 0, maka variabel *p* tidak berkontribusi dalam penentuan kelas. Kontribusi atribut atau variabel prediktor dapat dinilai dari besarnya nilai ***wi*** untuk masing-masing atribut, dengan *l*=1,2, ..., *p*.

Support Vector Machine (SVM) adalah suatu sistem pembelajaran yang menggunakan ruang hipotesis dari suatu fungsi linear dalam suatu ruang dimensi berfitur tinggi yang dikembangkan oleh Boser, Guyon, Vapnik, dan pertama kali dipresentasikan pada tahun 1992 di *Annual Workshop on Computational Learning Theory*.

Pada pendekatan *smoothing* yang digagas oleh Lee dan Mangasarian [6], kuadrat 2-*norm* dari vektor variabel *slack* ξ diminimalkan dengan bobot *v* / 2 menggantikan 1-*norm* dari vektor variabel *slack* ξ sehingga hasil modifikasi dari fungsi objektif SVM linier standar adalah sebagai berikut.

$$\min_{(w,\gamma,\xi)\in R^{n+1+m}} \frac{v}{2}\,\xi^T\xi + \frac{1}{2}(w^Tw + \gamma^2) \tag{2.22}$$

$$\text{s.t } D(Aw - e\gamma) + \xi \geq \text{e}, \xi \geq 0$$

Kendala pada Persamaan (2.22) dapat ditulis menjadi.

$$\xi = (e - D(Aw - e\gamma))_+ \tag{2.23}$$

di mana dalam hal ini $\text{x}_+ = \max\{0, x\}$. Bila Persamaan (2.23) disubtistusikan ke dalam persamaan (2.22) maka diperoleh fungsi objektif bebas kendala, yaitu.

$$\min_{(w,\gamma)\in R^{n+1}} \frac{v}{2}||\,(e - D(Aw - e\gamma))_+||_2^2 \quad + \frac{1}{2}(w^Tw + \gamma^2) \tag{2.24}$$

Bila dibandingkan dengan fungsi objektif (2.22), fungsi objektif (2.24) mereduksi dimensi dari yang semula $n + 1 + m$ menjadi $n + 1$. Namun fungsi objektif ini tidak memiliki turunan kedua sehingga metode optimasi konvensional tidak bisa digunakan. Oleh karena itu, Lee dan Mangasarian mengusulkan

smoothing technique, yaitu sebuah teknik di mana *plus function* x_+ pada fungsi objektif (2.39) didekati dengan sebuah fungsi *smooth-p* yang merupakan integral dari fungsi *signoid neural network* $1 + \exp(-\alpha x))^{-1}$ atau dapat ditulis sebagai berikut.

$$p(x, \alpha) = x + \frac{1}{\alpha}\ln(1 + \exp(-\alpha x)) \tag{2.25}$$

dengan $\alpha > 0$ adalah *smooth parameter* yang mengontrol kedekatan kurva $p(x, \alpha)$ terhadap kurva *plus function* x_+. Bila *plus function* x_+, pada Persamaan (2.24) diganti dengan $p(x, \alpha)$ maka diperoleh formulasi SSVM berikut.

$$\min_{(w,\gamma,\xi)\in R^{n+1}} \nabla\Psi_\alpha(w,\gamma) := \tag{2.26}$$

$$\min_{(w,\gamma)\in R^{n+1}} \frac{v}{2}|| \, p(e - D(Aw - e\gamma)) \quad , \alpha||_2^2 \quad + \frac{1}{2}(w^T w + \gamma^2)$$

Fungsi objektif SSVM (2.26) memiliki turunan kedua sehingga problem optimasi dapat dilakukan dengan menerapkan algoritma konvergen kuadrat Newton dengan tahapan Armijo yang membuat algoritma tersebut konvergen secara global.

Algoritma *Newton-Armijo* dimulai dengan menetapkan nilai awal $(w^0, \gamma^0) \in R^{n+1}$. Proses akan berhenti apabila gradien fungsi objektif SSVM, yaitu $\nabla\Psi_\alpha(w^i, \gamma^i) = 0$. Selain itu, (w^{i+1}, γ^{i+1}) akan dihitung dengan langkah – langkah berikut.

1. *Newton Direction*

   Menentukan *direction* $d^i \in R^{n+1}$ dengan menetapkan linierisasi sama dengan nol dari $\nabla\Psi_\alpha(w, \gamma)$ di sekitar (w^i, γ^i) yang memberikan $n + 1$ persamaan linier dengan $n + 1$ variabel.

   $$\nabla^2\Psi_\alpha(w^i, \gamma^i)d^i = -\nabla\Psi_\alpha(w^i, \gamma^i)' \tag{2.27}$$

2. Armijo *Sterpsize*

   Memilih sebuah *stepsize* $\lambda_i \in R$ sedemikian sehingga

   $$(w^{i+1}, \gamma^{i+1}) = (w^i, \gamma^i) + \lambda_i d^i \tag{2.28}$$

   dimana $\lambda_i = max\left\{1, \frac{1}{2}, \frac{1}{4}, \ldots\right\}$ sehingga

   $$\Psi_\alpha(w^i, \gamma^i) - \Psi_\alpha\left((w^i, \gamma^i) + \lambda_i d^i\right) \geq -\delta\,\lambda_i \nabla\Psi_\alpha(w^i, \gamma^i)d^i \tag{2.29}$$

dengan $\delta \in \left(0, \frac{1}{2}\right)$.

Saat $\Psi_\alpha(w^i, \gamma^i) = 0$, iterasi pada Algoritma *Newton-Armijo* berhenti diperoleh nilai w dan γ yang konvergen. Pada proses optimasi ini, matriks Hessian diberikan oleh Lee [6],

$$\lim_{a\to\infty} \nabla^2 \Psi_\alpha\,(w, y) = \begin{bmatrix} \frac{\partial^2 \Psi_a(w,\gamma)}{\partial^2 w^2} & \frac{\partial^2 \Psi_a(w,\gamma)}{\partial w \partial \gamma} \\ \frac{\partial^2 \Psi_a(w,\gamma)}{\partial^2 \partial w} & \frac{\partial^2 \Psi_a(w,\gamma)}{\partial^2 \gamma^2} \end{bmatrix} \tag{2.30}$$

$$= \mathrm{I} + \upsilon \begin{bmatrix} H_{1,1} & H_{1,2} \\ H_{2,1} & H_{2,2} \end{bmatrix}$$

di mana

$$H_{1,1} = A^T diag\left(S_\infty\big(e - D(Aw - e\gamma)\big)\right) A$$

$$H_{1,2} = -A^T diag\left(S_\infty\big(e - D(Aw - e\gamma)\big)\right) e$$

$$H_{2,1} = -e^T diag\left(S_\infty\big(e - D(Aw - e\gamma)\big)\right) A$$

$$H_{2,2} = Ae^T diag\left(S_\infty\big(e - D(Aw - e\gamma)\big)\right) e$$

dan

$$S_\infty(x) = \lim_{a\to\infty} \frac{1}{1 + \varepsilon^{-ax}} = \frac{1 + sign(x)}{2}$$

Sementara itu, gradien dari matriks Hessian memiliki formula sebagai berikut.

$$\lim_{a\to\infty} \nabla \Psi_\alpha\,(w, y) = \begin{bmatrix} w - vA^T D\big(\, e - D(Aw - ey)\big) \\ y + ve^T D\big(\, e - D(Aw - ey)\big) \end{bmatrix} \tag{2.31}$$

Nilai w dan γ optimum kemudian membentuk *classifier* dari SSVM Linier yang memiliki formulasi.

$$f(x) = sign(g(x)) \tag{2.32}$$

Dengan

$$g(x) = x^T w - \gamma \tag{2.33}$$

2.8 Bahasa Pemrograman Python

Python adalah Bahasa pemrograman bersifat umum. Python diciptakan pada tahun 1990 oleh Guido van Rossum. Bahasa level tinggi, stabil, dinamis, orientasi objek dan cross *platform* adalah karakteristik yang membuat Bahasa

python disukai oleh banyak pengembang. Tidak seperti bahasa lain yang susah untuk dibaca dan dipahami, python lebih menekankan pada keterbacaan kode agar lebih mudah untuk memahami sintaks. Hal ini membuat python sangat mudah dipelajari baik untuk pemula maupun untuk yang sudah menguasai Bahasa pemrograman lain. Bahasa pemrograman python berjalan dikebanyakan hardware dan system operasi, sehingga kebanyakan computer bias menjalankannya. Bahasa pemrograman Python saat ini dikembangkan dan dikelola oleh suatu tim relawan dengan nama *Python Software Foundation* [17]. Pembangunan program pada penelitian ini berbasis web menggunakan bahasa pemrograman python.

2.8.1 Jupyter Notebook

Jupyter Notebook (file yang berekstensi *ipynb*) adalah dokumen yang dihasilkan oleh *Jupyter Notebook App* yang berisikan kode komputer dan *rich text element* seperti paragraf, persamaan matematik, gambar dan tautan (*links*). *Jupyter Notebook* dikenal sebelumnya sebagai *IPython Notebook* dan dalam waktu dekat akan berevolusi menjadi *Jupyter Lab* [32]. Pada penelitian ini jupyter notebook digunakan untuk membangun rancangan code program berbentuk *console*.

2.8.2 Jetbrains Pycharm

IDE lain yang cukup populer dikalangan developer Python adalah PyCharm. PyCharm sendiri memiliki dua versi yaitu Professional Edition dan Community Edition. PyCharm Professional Edition merupakan versi berbayar dari PyCharm dan Community Edition merupakan versi gratis yang tersedia bagi komunitas python dengan lisensi Apache 2. *Pycharm* digunakan sebagai *tool* dalam membangun aplikasi dalam penelitian ini.

2.8.3 Flask

Flask adalah *microframework* untuk Python yang dibuat dengan *toolkit* wsgi dan Jinja2. Flask dibuat dan dimaintain oleh Armin Ronacher. Pertama kali dirilis 4 tahun yang lalu (April 2010), Flask saat ini berada di versi 0.10.1 dan dilisensikan dengan BSD license. Aplikasi web yang dibuat dengan Flask disimpan dalam satu berkas .py. Flask ingin menjadi *web framework* yang sederhana namun dapat diperluas dengan beragam pustaka tambahan yang sesuai dengan kebutuhan

penggunanya. Pada penelitian ini flask digunakan untuk membangun aplikasi web dan menyatukannya dengan source code program yang sudah dibuat di *pycharm*.

2.9 *Unified Modeling Language*

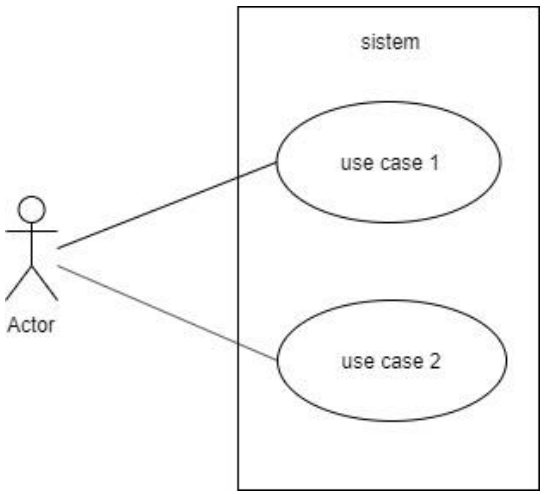
Unified Modeling Language (UML) adalah notasi yang lengkap untuk membuat visualisasi model suatu sistem. Sistem informasi dan fungsi, tetapi secara normal digunakan untuk memodelkan sistem computer. Didalam pemodelan objek guna menyajikan sistem yang berorientasi pada objek dan pada orang lain, akan sangat sulit dilakukan jika pemodelan tersebut dilakukan dalam bentuk kode Bahasa pemrograman [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. 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, use case scenario, activity diagram, class diagram,* dan *sequence diagram* [18].

2.9.1 *Use Case Diagram*

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

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 berprilaku kepada aktornya. Aktornya dalam *use case diagram* digambarkan debagai ikon yang berbentuk manusia yang biasanya dituliskan sebagai kata benda, sementara *use case* digambarkan sebagai ikon yang berbentuk elips yang berisi nama *use case* yang bersangkutan, biasanya dituliskan kata kerja untuk mempermudah pemahaman. Gambar 2.5 merupakan contoh *use case diagram.*



Gambar 2.3 Contoh *Use Case*

2.9.2 *Use Case Scenario*

Use case scenario merupakan penjelasan secara tekstual dari sekumpula skenario interaksi. Setiap skenario yang ada akan mendeskripsikan urutan aksi atau langkah yang dilakukan aktor ketika berinteraksi dengan sistem, baik dalam kondisi berhasil atau gagal [19].

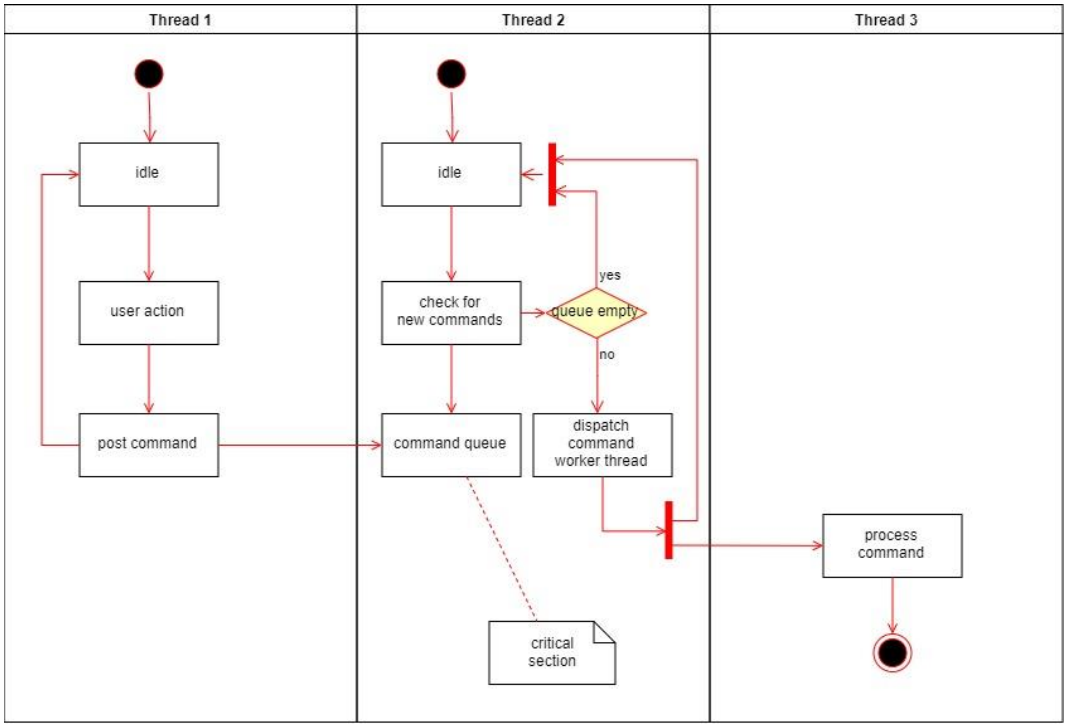
Use case scenario dijelaskan secara tekstual dalam beberapa format tergantung kebutuhannya, yaitu singkat (*brief), informal (casual),* atau lengkap (*fully dressed*), yang dapat dijelaskan falam satu atau dua kolom. Pada penelitian ini digunakan format yang diperkenalkan oleh Rules Miles [20] dengan bagian yang terlibat sebagai berikut :

1. Nama Use Case (*Use case name*)
2. Persyaratan Terkait (*Related Requirements*), kondisi spesifik yang harus terpenuhi sebelum use-case dieksekusi oleh aktor.
3. Tujuan (*Goal in Context*), penggunaan *use case* dan menjelaskan mengapa *use case* ini menjadi penting digunakan.
4. Komdisi Awal (*Precondition*), kondisi awal sebelum *use cse* dieksekusi.
5. Kondisi Akhir Berhasil (*Successful End Condition*), kondisi ketika *use cse* berhasil dieksekusi.
6. Kondisi Akhir Gagal (*Failed End Condition*), kondisi ketika *use case* gagal dieksekusi.
7. Aktor Utama (*Primary Actors*), Aktor utama yang menggunakan *use case.*

8. Aktor Sekunder (*Secondary Actors*), Aktor lainnta atau sekunder yang mengguanan *use case.*
9. Pemicu (*Trigger*), pemicu oelh aktor sehingga *use case* dieksekusi.
10. Alur Utama (*Main Flow*), penjelasan terkait alur utama dari *use case* apabila berjalan secara normal.
11. Ekstensi (*Ekstensions*), yaitu jalur alternatif dari interaksi yang terjadi antara aktor dari sistem yang mencakup percabangan (pilihan) maupun skenario yang gagal sehingga tujuan aktor tidak terpenuhi.

2.9.3 *Activity Diagram*

Activity diagram adalah diagram *flowchart* yang diperluas yang menunjukkan aliran kendali satu aktivitas ke aktivitas lain berdasarkan use case diagram [20]. Berikut adalah contoh dari activity diagram pada Gambar 2.7.



Gambar 2.4 Contoh *Activity Diagram*

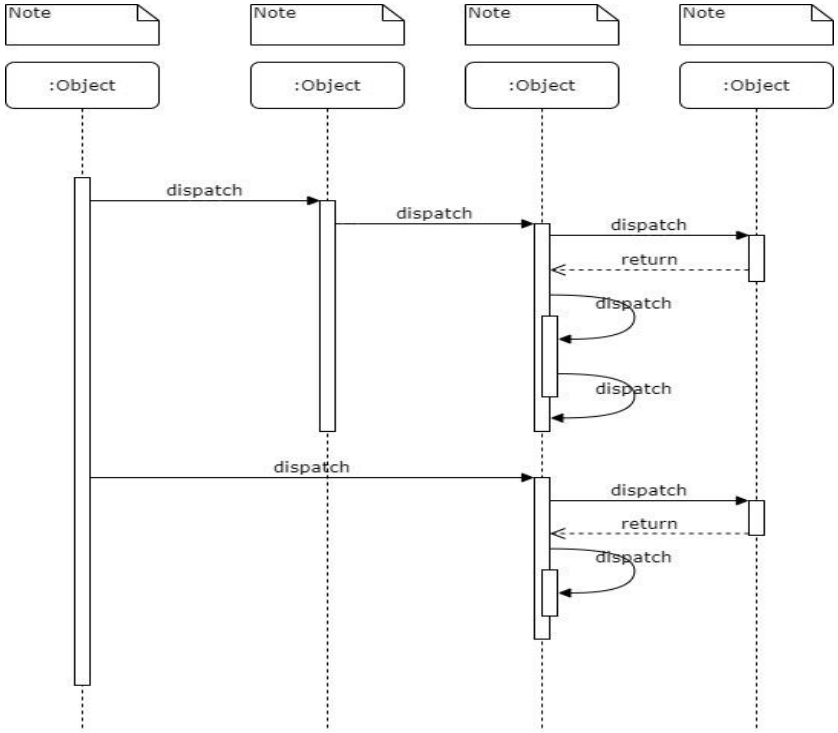
2.9.4 *Class Diagram*

Class diagram merupakan inti dari setiap sistem berorientasi objek, oleh karena itu kaitannya sangat erat dengan *use case diagram*. *Class diagram* bertujuan untuk menjelaskan berbagai jenis objek dan hubungannya dengan kelas lainnya

yang dimiliki oleh sistem. Hal inilah yang menjadikan *class diagram* memiliki dua macam informasi, yaitu informasi tentang kondisi awal objek dan bagaimana ia berprilaku dalam lingkungannya [20].

2.9.5 *Sequence Diagram*

Sequence diagram atau dikenal juga sebagai diagram interaksi bertujuan untuk memodelkan interaksi secara *runtime* antara bagian-bagian tertentu pada sistem yang membentuk alur perpindahan sebuah objek. Diagram ini akan menyampaikan urutan dari setiap interaksi pada suatu proses atau bagian-bagian sistem. Dengan menampilkan apa yang menjadi pemicu dari sebuah proses dan menunjukan informasi lain berupa peristiwa dalam suatu interaksi [20]. Berikut adalah contoh dari sequence diagram pada Gambar 2.8.



Gambar 2.5 Contoh *Squence Diagram*

2.10 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 [21].

Pengujian seharusnya meliputi tiga konsep berikut .

1) Demonstrasi validasi perangkat lunak pada masing-masing tahap diskusi pengembangan sistem.
2) Penentuan validitas sistem akhir dikaitkan dengan kebutuhan pemakai.
3) Pemeriksaan prilaku sistem dengan mengeksekusi sistem pada data sample 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 bagitu skesifikasi kebutuhan telah dapat didefinisikan. Evaluasi terhadap spesifikasi dan perancangan juga merupakan teknik pengujian. Kategori pengujian dapat dikategorikan menjadi dua [21], yaitu :

1. Berdasarkan ketersediaan *logic* sistem, terdiri dari *Black Box testing* dan *White Box testing.*
2. Berdasarkan arah pengujian, terdiri dari *top down* dan pengujian *Bottom up.*

2.10.1 Pengujian *Black Box*

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

2.10.2 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 [18] :

$$Akurasi = \frac{jumlah\ karakter\ sama}{jumlah\ seluruh\ karakter} \times 100\% \qquad (2.34)$$