BAB III

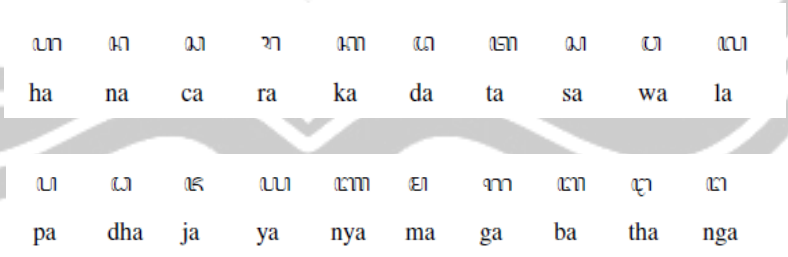
LANDASAN TEORI

3.1 Aksara Jawa

Aksara Jawa merupakan Aksara yang digunakan oleh masyarakat Jawa sejak jaman dulu. Aksara Jawa berkembang pada masa kejayaan kerajaan Majapahit. Ketika imperium Islam mulai memasuki tanah Jawa, format dan susunannya berubah seperti yang dikenali sekarang yang disebut *Hanacaraka* karena susunannya berbunyi *ha, na, ca, ra, ka*….dan seterusnya.

Aksara Jawa berdasarkan pada tata eja *Sriwedari* atau *weton Sriwedari* memiliki kelengkapan sebagai berikut.

1. Aksara *nglegena*, yang merupakan bentuk dasar Aksara karena belum mendapatkan imbuhan atau *sandhangan*. Gambar **3.1.1** menunjukkan ke-20 buah Aksara *nglegena*.



Gambar 3.1.1 Aksara *nglegena*

2. *Pasangan* Aksara *nglegena*, yang digunakan jika Aksara *nglegena* berada dibelakang Aksara yang bersifat *sigeg* (konsonan), yang proses perubahan konsonannya tidak dikarenakan mendapat *sandhangan sigeg pangkon*. Pasangan Aksara *nglegena* diletakkan di bawah Aksara yang

$$V_{nm}(\rho,\theta) = R_{nm}(\rho)e^{jq\theta} \text{……..(2)}$$

Keterangan :

n : bilangan integer bernilai positif atau 0

m : bilangan integer bernilai positif atau negatif, $|m| \leq n, n - |m|$ bernilai genap

ρ : Panjang vektor dari citra asal ke piksel (x, y); bernilai $\sqrt{x^2 + y^2}$

θ : Sudut antara vektor ρ dan sumbu x dalam arah berlawanan arah jarum jam yang bernilai $\tan^{-1}(\frac{y}{x})$

j : $\sqrt{-1}$

Radial polinomial bernilai real, didefinisikan sebagai :

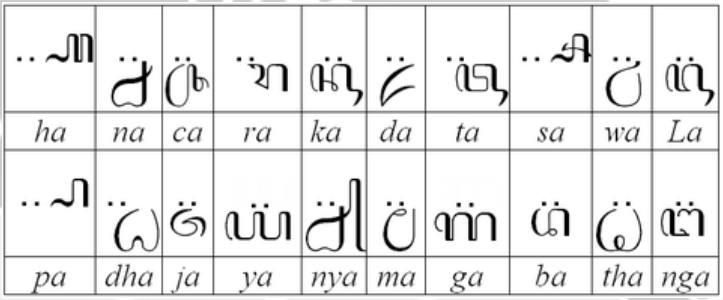
$$R_{nm}(\rho) = \sum_{k=0}^{n-|m|} (-1)^k \frac{(2n+1-k)!}{k!\,(n+|m|+1-k)!\,(n-|m|-k)} \rho^{n-k} \quad \text{...(3)}$$

dimana $0 \leq |m| \leq n$. Karena *Pseudo-Zernike Moment* di definisikan di koordinat polar (ρ, θ) dengan $|\rho| \leq 1$, maka komputasi polinomial *Pseudo-Zernike* membutuhkan transformasi linear dari citra dengan koordinat $(i, j), i, j = 0,1,2, \dots, N-1$ ke domain yang sesuai $x, y \in R^2$ didalam unit lingkaran.

Citra dapat direkonstruksikan dengan menerapkan transformasi invers sebagai:

$$\hat{f}(x_i, y_k) = \sum_{n=0}^{n_{max}} \sum_{m=-n}^{n} Z_{nm} V_{nm}(x_i, y_k), i, k = 0,1,2, \dots, N-1 \text{ … … … . . (4)}$$

dipasanginya kecuali untuk pasangan *ha, sa, pa,* dan *nya.* Sedangkan untuk pasangan *na* dan *wa* letaknya menggantung pada Aksara yang dipasanginya. Gambar 3.1.2 menunjukkan bentuk pasangan Aksara *nglegena*.



Gambar 3.1.2. Pasangan Aksara Nglegena
Sumber : http://pinterjawa.weebly.com/aksara-jawa.html

3. Aksara *Murda,* yang berjumlah delapan buah. Sebelumnya Aksara ini digunakan untuk menulis bunyi Aksara Jawa *Kuna*. Dalam perkembangannya Aksara *Murda* digunakan sebagai bentuk kapital dari huruf Jawa, yang digunakan untuk menulis nama, nama negara atau sesuatu yang dihormati.
4. Pasangan Aksara *Murda*, yang berfungsi untuk menggantikan Aksara *Murda* jika Aksara *Murda* tersebut berada dibelakang Aksara yang bersifat *sigeg* (konsonan), yang proses perubahan konsonannya tidak dikarenakan mendapat *sandhangan sigeg pangkon*.
5. Aksara *Rékan*, adalah Aksara yang digunakan untuk menulis ejaan huruf yang diadopsi dari kosakata bahasa Arab.
6. *Pasangan* Aksara *Rékan*, yang berfungsi untuk menggantikan Aksara *Rékan* apabila Aksara *Rékan* berada dibelakang Aksara yang bersifat *sigeg*

(konsonan), yang proses perubahan konsonannya bukan karena mendapat *sandhangan sigeg pangkon*.

7. Aksara *Swara*, yang berfungsi sebagai pelengkap untuk menuliskan kata-kata pinjaman dari bahasa asing seperti bahasa Inggris, Indonesia dan bahasa lainnya. Aksara *Swara* biasanya digunakan sebagai huruf kapital, Aksara ini tidak memiliki pasangan namun tetap bisa mendapatkan sandhangan utamanya yaitu sandhangan *sigeg*. Jumlah Aksara Swara ada lima buah yang dalam tulisan Latin disebut dengan A I U E O.
8. Angka, yang digunakan untuk menuliskan angka yang dikenal saat ini.
9. *Sandhangan*, yang merupakan simbol-simbol tambahan untuk mengubah bunyi Aksara, seperti terdapat pada Gambar 3.1.3. *Sandhangan* terbagi dalam tiga jenis, yaitu *Sandhangan Swara, Sigeg,* dan *Anuswara.*

Nama Sandhangan	Aksara Jawa	Keterangan	Nama Sandhangan	Aksara Jawa	Keterangan
Wulu	◌	tanda vokal i	Wignyan	◌	tanda ganti konsonan h
Suku	◌	tanda vokal u	Cecak	◌	tanda ganti konsonan ng
Taling	◌	tanda vokal é	Pangkon	◌	tanda penghilang vokal
Pepet	◌	tanda vokal e	Péngkal	◌	tanda ganti konsonan ya
Taling Tarung	◌	tanda vokal o	Cakra	◌	tanda ganti konsonan ra
Layar	◌	tanda ganti konsonan r	Cakra keret	◌	tanda ganti konsonan re

Gambar 3.1.3 Bentuk Sandhangan

Sumber : http://pinterjawa.weebly.com/aksara-jawa.html

10. *Adeg-adeg* atau tanda baca.

3.2 Optical Character Recognition (OCR)

Optical Character Recognition (OCR) atau pengenalan karakter optikal merupakan salah satu fokus penelitian dalam pengenalan pola citra digital. Pada dasarnya OCR merupakan aplikasi atau sistem untuk memindai tulisan pada gambar untuk kemudian diubah kedalam bentuk teks yang dapat dimanipulasi.

3.3 Pengolahan Citra Digital

Pengolahan citra digital menurut Jain(1989) adalah "pemrosesan gambar berdimensi-dua melalui komputer digital". Pengolahan citra digital umumnya terbagi dalam dua jenis kegiatan, yaitu memperbaiki kualitas gambar dan mengolah informasi pada gambar untuk keperluan pengenalan objek.

Pada penelitian ini, terdapat beberapa langkah pengolahan citra yang disebut dengan *preprocessing*, yaitu proses pengolahan citra agar citra tersebut untuk mengambil informasi yang diperlukan dalam proses klasifikasi objek yang dikenali. Langkah-langkah pengolahan citra yang digunakan tersebut yaitu *grayscalling*, *thresholding*, *resizing*, *erosion,* dan ekstraksi fitur.

1. *Grayscalling*

   Grayscalling adalah operasi mengubah citra berwarna menjadi citra berskala keabuan.

2. *Thresholding*

   Thresholding merupakan operasi pengubah suatu citra menjadi citra biner. Citra biner adalah citra yang setiap pikselnya hanya bernilai 1 atau 0,

3. *Erosion*

   Erosion atau pengikisan merupakan operasi morfologi untuk memperkecil tepi dari suatu objek citra.

4. *Resizing*

   Resizing adalah proses mengubah ukuran tinggi dan lebar citra kedalam ukuran tertentu.

5. Ekstraksi fitur

   Ekstraksi fitur merupakan pengambilan ciri dari suatu objek. Dalam penelitian ini, penulis menggunakan algoritma *Pseudo Zernike Moment* untuk melakukan ekstraksi fitur. Fitur yang diekstraksi akan digunakan pada tahap klasifikasi.

3.4 Pseudo-Zernike Moment

Pseudo-Zernike Moment digunakan dalam beberapa aplikasi pengenalan pola untuk ekstraksi fitur pada citra(Chong, Raveendran & Mukundan, 2003). Metode ini telah terbukti lebih unggul dari fungsi lain seperti *Zernike Moment* konvensional dalam hal kemampuan representasi fitur dan ketahanannya dalam kesalahan kuantisasi gambar dan noise(Kef, Chergui & Chikhi, 2016).

Ekstraksi fitur *Zernike* mendefinisikan rangkaian ortogonal kompleks dari polinomial kompleks di atas ruang koordinat polar di dalam lingkaran unit, misalnya $x^2 + y^2 = 1$. Momen *Pseudo-Zernike* dua dimensi ordo *n* dengan pengulangan *m* untuk fungsi gambar kontinyu $f(x, y)$ didefinisikan sebagai :

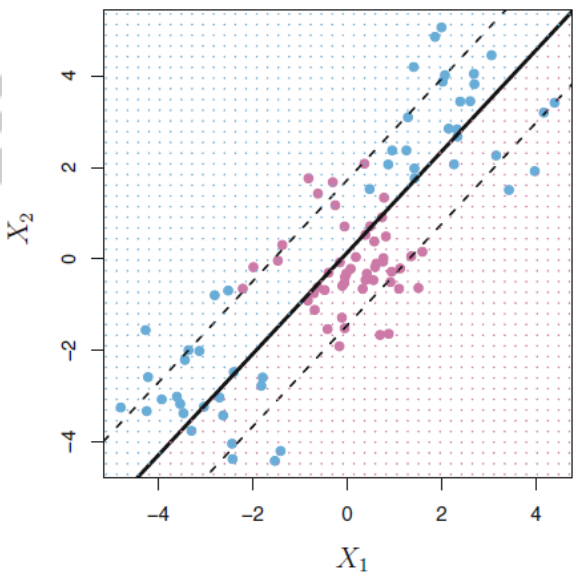
$$Z_{nm} = \frac{n+1}{\pi} \sum_{x} \sum_{y} f(x, y)\, V_{nm}^{*}(\rho, \theta), \qquad x^2 + y^2 \leq 1 \qquad \text{.....(1)}$$

dimana * menunjukkan konjugat kompleks, dan $V_{nm}(\rho, \theta)$ adalah polinomial *Pseudo-Zernike* ordo p dengan pengulangan q diberikan oleh

Seluruh momen Z_{nm} dari *f(x,y)* sampai ordo maksimal n_{max} tersebut dapat digunakan sebagai vektor fitur.

3.5 *Support Vector Machine* (SVM)

Support Vector Machine (SVM) adalah metode klasifikasi yang berfungsi untuk memisahkan dua buah kelas secara linier dengan mencari margin terbaik pada suatu bidang pemisah (*hyperplane*) untuk memisahkan kedua kelompok kelas. SVM merupakan perpanjangan dari *Support Vector Classifier* yang dihasilkan dari memperluas *feature space* dengan cara tertentu, menggunakan kernels(James et al., 2014). Ide utamanya adalah kita ingin memperluas *feature space* untuk mengakomodasikan non-linear boundary diantara kelas. Pada praktiknya, tidak semua *dataset* dapat dipisahkan secara linier. Seperti terdapat pada Gambar 3.5.1, data yang terbagi dalam dua kelas tersebut tidak bisa dipisahkan secara linier.



Gambar 3.5.1. Dua buah kelas yang tidak bisa dipisahkan secara linier

Salah satu cara untuk mengatasi masalah ketidaklinieran adalah dengan mentransformasikan data kedalam *feature space* agar data dapat dipisahkan secara linier. Terdapat dua hal yang dapat menjadi masalah saat mentransformasikan data kedalam *feature space*. Mentransformasikan data kedalam *feature space* membutuhkan komputasi yang sangat besar karena *feature space* bisa saja memiliki jumlah *feature* yang tak terhingga. Masalah lain yaitu sulit untuk mengetahui fungsi transformasi yang tepat. Karakteristik penting dari optimasi SVM adalah bahwa semua operasi dapat dimasukkan kedalam *inner product* (Theodoridis et al., 2010). Jadi, untuk menyelesaikan masalah linear dalam ruang berdimensi tinggi (setelah pemetaan), yang harus kita lakukan adalah mengganti inner product kedalam bentuk $K(x_i, x_{i'})$, dimana K adalah fungsi yang disebut *kernel*. Kernel adalah fungsi yang mengukur kemiripan dua pengamatan(James et al., 2014). Dengan demikian, fungsi yang dihasilkan dari pelatihan menjadi :

$$f(x) = \sum_{i \in S}^{N} \alpha_i y_i K\langle x, x_i \rangle + \beta_0 \quad \text{.....(5)}$$

Salah satu fungsi *kernel* yang populer digunakan adalah *kernel* radial basis, yang memiliki bentuk :

$$K(x_i, x_{i'}) = \exp(-\gamma \|x - x'\|^2) \text{.....(6)}$$

Seiring dengan perkembangan permasalahan yang hendak diselesaikan metode SVM saat ini dapat digunakan untuk dapat mengklasifikasikan data yang memiliki kelas lebih dari dua. Terdapat banyak metode yang digunakan untuk mengimplementasikan *multi-class* SVM, salah satunya yaitu *one-against-all*.

Pada metode *one-agaist-all* terdapat *hyperplane* sebanyak jumlah kelas. Setiap *hyperplane* bertugas membedakan apakah setiap data masuk ke suatu kelas tertentu atau atau tidak. Sebuah data akan dibandingkan pada *hyperplane* pertama, apakah termasuk dalam kelas pertama atau tidak. Jika tidak, maka dilanjutkan dengan *hyperplane* selanjutnya. Bagitu seterusnya hingga *hyperplane* terakhir.