# Sistem Pengenalan Aksara Sunda Menggunakan Metode *Modified Direction Feature* Dan *Learning Vector Quantization*

Rizki Rahmat Riansyah[#1], Youllia Indrawaty Nurhasanah[#2], Irma Amelia Dewi[#3]

[#]*Jurusan Teknik Informatika, Institut Teknologi Nasional Bandung*
*Jl. PHH. Mustafa No. 23, Bandung*

[1]er.rizkiansyah@gmail.com

[2]youllia@itenas.ac.id

[3]imameamel@gmail.com

*Abstract* — **Sundanese script is one local Indonesian script which has the specificity in terms of how the writing system by using a non-latin character as well as in terms of the unique pronunciation, therefore that presence should be conserved. One of its preservation are to build an sundanese recognition system. This system will accept input in the image of sundanese script that will be processed in order to generate output in the form of text. In the process of recognizing a pattern from the image of sundanese script can use the techniques of extraction feature, one of which is modified direction feature (MDF), whereas the methods used for classification and identification process is learning vector quntization (LVQ). To support it's function of this image recognition system combined with text to speech (TTS). TTS system is a system that can convert text into speech, so the output of this system can display text results of sundanese recognition script along with the examples of pronunciation. The level of accuracy of the test results of the 300 samples data image of sundanese script verified correctly between the suitability of image characters with the names and the pronunciation is 78.67%.**

*Keywords*— **Pattern Recognition, Feature Extraction, Classification, Text-to-Speech.**

## I. PENDAHULUAN

### A. Latar Belakang

Aksara Sunda merupakan salah satu aksara daerah Indonesia yang merupakan hasil karya ortografi masyarakat Sunda melalui perjalanan sejarahnya sejak 5 abad yang lalu hingga saat ini. Aksara Sunda sejak tahun 2008 sudah resmi menjadi bagian standar *Unicode*[14].

Oleh karena itu, dalam perkembangannya naskah maupun aksara sunda yang dicetak menggunakan *font* standar aksara sunda akan banyak ditemukan[4], sehingga dapat dikembangkan ke dalam kegiatan-kegiatan lainnya yang berkaitan dengan komputerisasi, salah satunya adalah pengenalan pola citra.

Pengenalan pola (*pattern recognition*) merupakan salah satu aplikasi jaringan syaraf tiruan untuk mengenali suatu pola (misal huruf, angka, suara atau tanda tangan) [18], baik tulisan dengan aksara Latin maupun aksara *non*-Latin. Salah satu contoh tulisan aksara *non-latin* adalah aksara sunda yang memiliki kekhasan yaitu cara penulisan yang mengunakan sistem karakter *non-latin* serta dari segi cara pengucapannya yang unik.

Pada umumnya teknik yang digunakan untuk mengenali pola citra aksara diperlukan teknik ekstraksi ciri dan pengelompokan pola. Dalam prosesnya teknik yang dapat digunakan salah satunya adalah *modified direction feature* (MDF) dimana metode ini merupakan metode ekstrasi fitur yang mentransformasi vektor dari citra biner menjadi sebuah vektor yang mewakili ciri dari arah dan transisi dari suatu pola. Selanjutnya teknik yang digunakan adalah *learning vector quantization* (LVQ) dimana metode ini digunakan sebagai metode pengelompokan atau pengklasifikasi dari suatu citra. Untuk menunjang fungsinya pada pengenalan aksara ini dapat dikombinasikan dengan teknik *text to speech*. Sistem *text to speech* atau TTS merupakan sebuah sistem yang dapat mengubah teks menjadi ucapan[10].

Oleh karena itu, dalam penelitian ini dirancang sebuah aplikasi yang dapat mendeteksi atau mengenali aksara sunda. Cara kerja sistem ini adalah dengan memanfaatkan sebuah kamera atau *scanner* dimana sistem akan menerima *input* dokumen berupa gambar, yang kemudian akan diproses oleh sistem dengan metode MDF dan LVQ, sehingga *output* dapat menampilkan pengenalan aksara sunda dan contoh pengucapannya.

### B. Rumusan Masalah

Rumusan masalah pada aplikasi sistem pengenalan aksara sunda ini adalah sebagai berikut :

1. Bagaimana teknik ekstraksi ciri dengan menggunakan metode MDF ?
2. Bagaimana proses klasifikasi ciri dengan menggunakan metode LVQ ?

3. Bagaimana cara sistem agar dapat menampilkan *output* berupa *speech* ?
4. Seberapa besar tingkat akurasi pengenalan aksara sunda dengan menggunakan metode MDF dan LVQ ?

### *C. Tujuan*

Tujuan dari penelitian ini adalah membangun sistem pengenalan aksara sunda dengan menerapkan metode MDF dan LVQ sebagai proses untuk mengenali aksara sunda.

### *D. Batasan Masalah*

Batasan masalah yang dibatasi pada penelitian ini, diantaranya :

1. *Input* merupakan *image.*
2. *Background input* putih.
3. Pola *input* dengan posisi tegak simetris.
4. Pengenalan citra ditunjukan pada satu aksara huruf dasar sunda.
5. Pengenalan citra ditunjukan pada aksara ngalagena dan swara.
6. Ukuran citra aksara sunda yang diteliti yaitu 20pt, 72pt, dan 150pt.
7. Ukuran piksel kamera yang digunakan adalah 13mp dan 16mp.
8. Data yang akan diuji sudah disiapkan terlebih dahulu pada *drive user*.

## II. METODOLOGI PENELITIAN

### *A. Subjek Penelitian*

Subjek penelitian ini adalah citra dengan teknik pengambilan yang berbeda. Citra tersebut diambil dengan menggunakan *camera*, *scanner* dan *screenshot text* dengan perbedaan ukuran aksara yaitu 20pt, 72pt, dan 150pt. Citra tersebut memiliki karakteristik yang berbeda dimana setiap citra mewakili 1 huruf dasar aksara sunda.

### *B. Teknik Pengumpulan Data*

Pengujian penelitian ini dilakukan dengan mengambil data citra melalui *camera*, *scanner* dan *screenshots text* dengan ukuran pada aksara sunda 20pt, 72pt, dan 150pt pada setiap citra aksara sunda. Setiap pengujian satu citra aksara dilakukan sebanyak 5 kali percobaan. Jika dalam 3 kali percobaan sesuai maka aksara tersebut dianggap cocok/dapat dikenali. Tetapi jika hanya 2 kali percobaan yang sesuai maka dianggap gagal/tidak dapat dikenali.

### *C. Sudi Literatur*

Literatur yang digunakan adalah yang terkait dengan metode *modified direction feature, learning vector quantization,* dan *teks to speech*. Pembelajaran tersebut dilakukan dengan cara mencari referensi dan pengumpuan data yang diperoleh dari buku, jurnal, artikel, dan *website* terkait yang telah dilakukan sebelumnya.

Berdasarkan perbandingan tinjauan pustaka, penulis terinspirasi untuk melakukan penelitian mengenai pengenalan aksara sunda dan pengenalan teks ke ucapan seperti penelitian yang telah dilakukan oleh **Hendro[8]** dengan judul **"Konversi Alfabet Latin ke Aksara Sunda Kaganga"**, didapat bahwa Algoritma pencocokan *string* dapat diterapkan pada aksara sunda, perbedaannya pada penelitian **Hendro** adalah mengkonversi alphabet latin ke aksara sunda, sedangkan pada penelitian sistem pengenan aksara sunda adalah mengenali bentuk aksara non-latin menjadi sebuah alphabet. Lalu untuk metode ekstraksi ciri yang digunakan dalam penelitian ini adalah MDF sama seperti penelitian yang digunakan peneliti kedua oleh **Gilang, dkk[7]** dengan penelitiannya mengenai pengenalan karakter optik menggunakan metode MDF, peneliti ketiga oleh **Erik, dkk[4]** yang melakukan penelitian untuk pengenalan aksara sunda dengan metode MDF dan JST RBF, dan peneliti keempat oleh **Tjokorda, dkk[19]** dalam penelitiannya yang berjudul **"Pengenalan Huruf Bali menggunakan metode MDF dan LVQ"**, dengan tingkat akurasi keberhasilan pengujian berdasarkan penelitian yang telah dilakukan ketiga peneliti tersebut mengenai pengenalan karakter optik untuk karakter *non*-latin dengan menggunakan metode MDF sebagai ekstraksi cirinya memiliki tingkat akurasi keberhasilan di atas 68% sehingga metode MDF dapat digunakan sebagai metode ekstraksi ciri pada citra dengan karakter *non*-latin karena metode MDF memiliki kelebihan yaitu ciri dari suatu karakter yang dihasilkan memiliki nilai pembeda yang tinggi karena metode ini merupakan gabungan dari dua buah metode yaitu *direction feature* (DF) dan *transition feature* (TF), perbedaan dari ketiga penelitian tersebut adalah proses klasifikasi dan identifikasi yang digunakan pada sistem pengenalan aksara sunda menggunakan metode LVQ. Metode LVQ sudah banyak digunakan pada berbagai penelitian sama seperti penelitian yang dilakukan peneliti kelima oleh **Maulana, dkk[11]** mampu membangun sistem pengenalan huruf hijaiyah dengan menggunakan metode *Chain Code* dan LVQ, peneliti keenam oleh **Fachrul dan Hani [5]** dengan penelitiannya membangun sistem pengenalan tulisan menggunakan metode LVQ, dan peneliti ketujuh oleh **Dea, dkk[3]** yang melakukan penelitian dengan membangun pengenal kata dalam aksara sunda, berdasarkan ketiga penelitian tersebut dapat disimpulkan bahwa pengenalan aksara *non*-latin dengan menggunakan metode LVQ sebagai metode pengklasifikasi mampu direalisasikan dengan tingkat akurasi diatas 60%, perbedaan dari ketiga penelitian tersebut terletak pada metode ekstraksi ciri serta tidak adanya proses *text to speech*. Untuk menunjang fungsinya pada penelitian sistem pengenalan aksara sunda ini menggunakan TTS system untuk dapat mengubah teks menjadi ucapan seperti yang telah dilakukan peneliti kedelapan oleh **Rd. Rakha, dkk[9]** dengan penelitiannya yang membandingkan metode *Diphone Concatenation* dan Algoritma *Sonic* pada sistem TTS. Peneliti kesembilan oleh **Abi dan Riana[1]** dan peneliti

kesepuluh oleh **Dannis[2]** yang melakukan penelitian dengan menggunakan algoritma KMP sebagai metode pencarian *string* untuk sistem TTS mengenai pembuatan TTS sederhana menggunakan algoritma KMP, ketiga penelitian tersebut mampu diperoleh sebuah aplikasi yang dapat melakukan konversi teks kedalam bentuk ucapan dengan menggunakan permodelan bahasa manusia, perbedaannya adalah ketiga penelitian tersebut tidak menggunakan proses pengolahan citra, sedangkan pada sistem pengenalan aksara sunda menggunakan pengolahan citra terlebih dahulu untuk mendapatkan *output* berupa teks.

Oleh karena itu, penelitian yang berjudul **"Sistem Pengenalan Aksara Sunda Menggunakan Metode *Modified Direction Feature* dan *Learning Vector Quantization*"** ini mengaplikasikan metode MDF untuk proses ekstraksi ciri, sedangkan untuk proses klasifikasi menggunakan metode LVQ dan untuk pen-sintesis teks ke ucapan menggunakan algoritma KMP.

## III. LANDASAN TEORI

### A. *Pengenalan Pola*

Sistem pengenalan pola pada dasarnya meliputi tiga tahap berrikut : (i) akuisisi data, (ii) pra-pengolahan data, dan (iii) pembuatan keputusan [13].

*Image pre-processing* adalah tahap menjadikan citra menjadi biner sebelum citra diproses pada tahap ekstraksi ciri. Pada tahap *image pre-processing*, dapat dibedakan menjadi 3 tahap, yaitu *Resize* merupakan proses mengubah ukuran citra agar citra yang diproses memiliki ukuran piksel yang sama. *Grayscale* berfungsi untuk mengubah citra berwarna menjadi citra yang hanya memiliki derajat keabuan. *Thinning* yang bertujuan untuk mendapatkan garis pada citra huruf menjadi 1 piksel dengan proses penipisan citra huruf.

### B. *Metode Modified Direction Feature (MDF)*

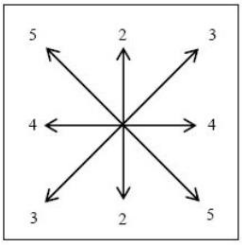
MDF adalah sebuah metode ekstraksi ciri pada citra digital dengan menggabungkan dua buah metode yaitu metode *Direction Feature* (DF) dan metode *Transition Feature* (TF)[7],[19].

*1) Direction Feature (DF) :* DF adalah pencarian nilai feature berdasarkan label arah dari sebuah *pixel*. Pada metode ini setiap *pixel foreground* (piksel karakter) pada citra digital akan diberi 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 dapat dilihat pada Tabel 1 dan Gambar 1.

TABEL I

NILAI LABEL DAN ARAH PADA DF

| Arah | Nilai | Bentuk |
|---|---|---|
| Vertikal | 2 | |
| Diagonal Kanan | 3 | |
| Horizontal | 4 | |
| Diagonal Kiri | 5 | |




Gambar 1. Pelabelan arah pixel pada DF
(Sumber : Jurnal Penelitian Analisis dan Implementasi *Optical Character Recognition* Menggunakan Metode *Modified Direction Feature* dan *Least Squares Support Vector Machine*, Gilang : 2013 )

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

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 Tabel 2. Misalkan O adalah sebuah *foreground*, maka nilai arah O didapatkan dengan melakukan pengecekan secara berurutan dari X1 – X8. Pengecekan X dihentikan jika sudah ditemukan X pertama yang merupakan *foreground*, maka ubahlah nilai O menjadi nilai arah berdasarkan aturan dibawah ini:
   - Jika pada posisi X1 atau X5 maka nilai arah adalah 5
   - Jika pada posisi X2 atau X6 maka nilai arah adalah 2
   - Jika pada posisi X3 atau X7 maka nilai arah adalah 3
   - Jika pada posisi X4 atau X8 maka nilai arah adalah 4

TABEL II

MATRIX KETETANGGAAN PENENTUAN NILAI LABEL

| | | |
|---|---|---|
| X1 | X2 | X3 |
| X8 | O | X4 |
| X7 | X6 | X5 |

*2) Transition Feature (TF) :* TF adalah sebuah metode untuk menghitung posisi transisi dan jumlah transisi pada bidang vertikal dan horizontal dari suatu gambar. Transisi adalah posisi dimana terjadinya perubahan pixel dari background menjadi foreground tetapi tidak sebaliknya, dilakukan secara transversal dari empat arah, yaitu dari kanan ke kiri, kiri ke kanan, atas ke bawah, dan bawah ke atas. Nilai *longitude transition* (LT) pada TF didapat dari pembagian antara posisi transisi dengan panjang ataupun lebar dari suatu gambar. Berikut adalah persamaan 1 perhitungan nilai LT untuk pemindaian dari arah kiri ke kanan dan dari atas ke bawah menggunakan persamaan 1:

$$LTi = 1 - (\frac{xi}{Maxi}) \quad \text{.....(1)}$$

Sedangkan perhitungan nilai LT untuk pemindaian dari arah kanan ke kiri dan dari bawah ke atas dengan menggunakan persamaan 2 :

$$LTi = (\frac{xi}{Maxi}) \quad \text{.....(2)}$$

Dimana $xi$ adalah indeks pixel yang dikaji dihitung dari awal pencarian dan $Maxi$ merupakan jumlah *pixel* maksimal dalam satu baris atau kolom segmen pixel citra mengikuti aturan berikut :

- Jika pemindaian dilakukan dari kiri ke kanan maka nilai $Maxi$ adalah lebar citra.
- Jika pemindaian dilakukan dari atas ke bawah maka nilai $Maxi$ adalah panjang citra.
- Jika pemindaian dilakukan dari kanan ke kiri maka nilai $Maxi$ adalah lebar citra.
- Jika pemindaian dilakukan dari bawah ke atas maka nilai $Maxi$ adalah panjang citra.

Nilai LT dari masing-masing arah akan selalu berkisar antara 0-1. LT sangat dipengaruhi oleh jumlah transisi yang digunakan dimana jumlah transisi yang bisa dicatat dalam 1 baris atau 1 kolom citra. Jumlah transisi yang diambil dari setiap arah tergantung dari jumlah transisi maksimal yang ditetapkan. Apabila terdapat transisi lebih dari jumlah maksimal transisi maka transisi tersebut tidak akan dihitung. Namun apabila jumlah transisi yang ditemukan kurang dari jumlah maksimal maka nilai transisi sisanya yang diberikan adalah 0.

Setelah nilai TF ditemukan dari masing-masing LT yang telah dicari, perhitungan *direction transition* (DT) pada DF dilakukan untuk setiap transisi yang terjadi pada TF, jadi jika suatu transisi pada TF ditemukan, maka DF untuk *foreground* yang bersangkutan juga dihitung. Sehingga akan didapatkan pasangan [TF,DF] setiap kali ditemukan transisi pada TF dari keempat arah. Nilai DT dari DF pada MDF diambil dari pembagian antara nilai DF dengan 10. Nilai 10 diambil untuk mendapatkan rentang nilai antara 0-1. Persamaan 3 merupakan perhitungan nilai DT :

$$DTi = (\frac{Nilai\_Arah\,(DF)\_i}{10}) \quad \text{.....(3)}$$

Dengan adanya perhitungan matriks LT dan DT yang dihitung dari keempat arah pencarian, maka dilakukan normalisai vektor ciri dengan menggunakan persamaan 4, sehingga akan dihasilkan vektor sejumlah :

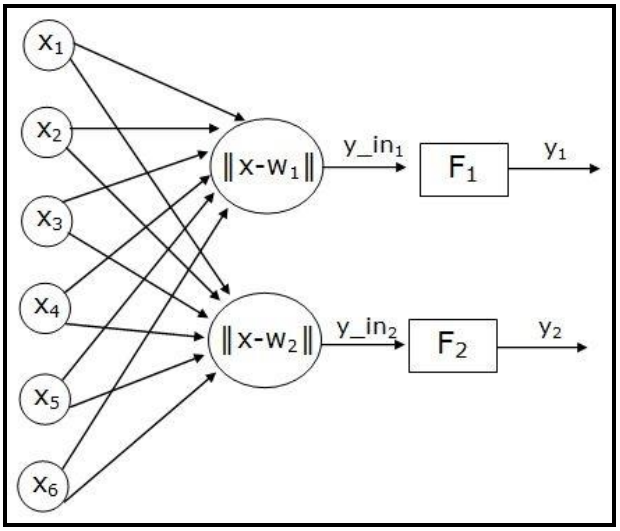
$$normalisasi = nrFeature\ x\ nrTrantition\ x\ nrVektor\ x\ nrMatrixHeight(Width) \quad \text{.....(4)}$$

dimana :

$nrFeature$ = 2 (LT dan DT)
$nrTrantition$ = Jumlah transisi yang dipakai
$nrVektor$ = 4 (jumlah arah pencarian)
$nrMatrixHeigh$ = 5 (jumlah ukuran normalisasi matriks)

*C. Metode Learning Vector Quantization (LVQ)*

LVQ adalah suatu metode untuk melakukan pembelajaran pada lapisan kompetitif yang terawasi. Suatu lapisan kompetitif akan secara otomatis belajar untuk mengklasifikasi vektor-vektor input. Kelas-kelas yang didapatkan sebagai hasil dari lapisan kompetitif ini hanya tergantung pada jarak antara vektor-vektor input. Jika dua vektor *input* mendekati sama, maka lapisan kompetitif akan meletakkan kedua vektor input tersebut ke dalam kelas yang sama. Contoh arsitektur LVQ terlihat pada `Gambar 2` [11].




Gambar 2. Arsitektur LVQ
( Sumber : Jurnal Penelitian Implemetasi Algoritma Chain Code Dan Lvq Pada Pengenalan Huruf Hijaiyah, Maulana : 2016 )

Penghitungan jarak *Euclidean* antara vektor input dan vektor bobot dilakukan dengan menghitung nilai *Euclidean Distance* menggunakan persamaan 5.

$$C_j = \sqrt{\sum_{i=1}^{n} (x_i - w_j)^2} \quad \text{.....(5)}$$

dimana :

$C_j$ adalah nilai jarak euclidean
$x_i$ adalah nilai vektor input
$w_j$ adalah nilai vektor bobot

Selain itu LVQ selalu memperbaharui bobotnya sesuai jumlah maksimal epoch yang ditetapkan. Epoch yaitu satu siklus pelatihan yang melibatkan semua pola. Persamaan 6 merupakan rumus bobot LVQ yang digunakan untuk memperbaharui nilai bobot.

$$\vec{W_m} \leftarrow \vec{W_m} + \alpha \cdot (\vec{X} - \vec{W_m}) \quad \text{.....(6)}$$

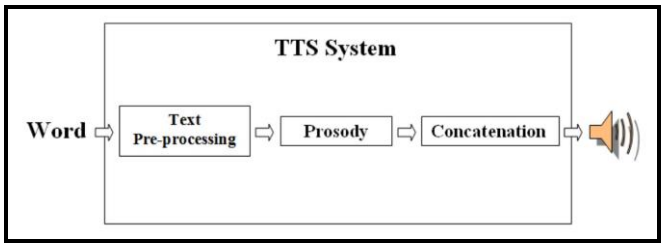
dimana :

$\vec{W_m}$ adalah bobot
$\alpha$ adalah *learning rate*
$\vec{X}$ adalah input

*D. Text To Speech (TTS)*

Pada dasarnya TTS adalah suatu sistem yang dapat mengubah teks menjadi ucapan.

*Text-to-Speech* adalah suatu sistem yang dapat mengubah teks menjadi ucapan. Menurut Joko [10] suatu pensintesa ucapan atau *Text to Speech synthesis system* terdiri dari 3 bagian, yaitu *text pre-processing*, pembangkitan *prosody* dan *concatenation*. Gambar 3 adalah diagram blok *text to speech synthesis system* :




Gambar 3. Blok diagram TTS
(Sumber : Naskah publikasi Rancang Bangun Aplikasi *Text-to-Speech* Sebagai Alat Bantu Pembelajaran Bahasa Inggris, Joko : 2013)

*E. Knuth-Morris-Pratt (KMP)*

KMP adalah suatu algoritma pencarian *string* yang diciptakan oleh Donald Knuth dan Vaughan Pratt, dan secara independen oleh J.H.Morris pada 1977, namun dipublikasikan secara bersamaan[2].

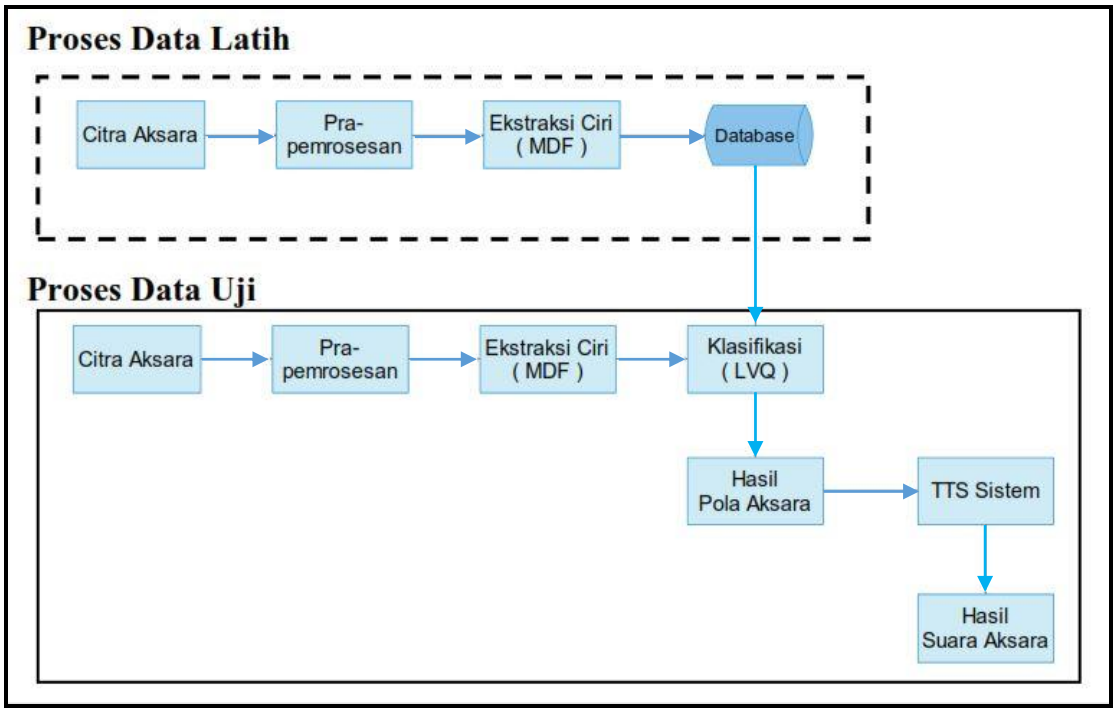
Secara sistematis, langkah-langkah yang dilakukan algoritma *Knuth-Morris-Pratt* pada saat mencocokkan *string* :

1. Masukkan *Query* kata yang akan dicari. Dengan permisalan
   P=*Pattern* atau pola susunan kata yang dijadikan sebagai contoh atau pola teks yang akan dicari
   T=Teks atau judul dokumen
2. Algoritma KMP mulai mencocokkan *pattern* atau pola susunan kata yang dijadikan sebagai contoh pada awal teks.
3. Dari kiri ke kanan, algoritma ini akan mencocokkan karakter per karakter *pattern* atau pola susunan kata yang dijadikan sebagai contoh dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi :
   - Karakter di *pattern* atau pola susunan kata yang dijadikan sebagai contoh dan di teks yang dibandingkan tidak cocok (*mismatch*).
   - Semua karakter di *pattern* atau pola susunan kata yang dijadikan sebagai contoh cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
4. Algoritma kemudian menggeser *pattern* atau pola susunan kata yang dijadikan sebagai contoh berdasarkan tabel next, lalu mengulangi langkah no. 2 sampai *pattern* atau pola susunan kata yang dijadikan sebagai contoh berada di ujung teks.

## IV. ANALISIS DAN PEMBAHASAN

*A. Proses Kerja Sistem Pengenalan Aksara Sunda*

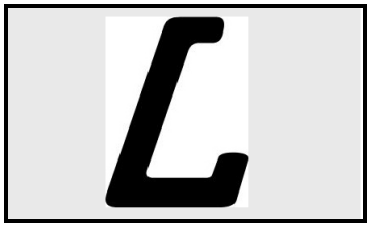
Proses kerja dari dari sistem pengenalan aksara sunda pada aplikasi ini digambarkan dalam bentuk blok diagram seperti yang terlihat pada Gambar 4.




Gambar 4. Blok diagram aplikasi

Dalam proses data latih, *input*an merupakan citra aksara, lalu citra tersebut akan dilakukan tahap *pre-processing*, selanjutnya untuk mendapatkan nilai ciri dari sebuah citra dilakukan proses ekstraksi ciri, nilai ciri yang diperoleh akan disimpan ke dalam *database* untuk dilakukan proses pelatihan. Proses selanjutnya pada data uji, *input*an merupakan citra aksara yang akan dilakukan tahap *pre-proccessing*, lalu ekstraksi ciri, kemudian dilakukan proses klasifikasi yang merupakan proses pengelompokan seluruh *pixel* pada suatu citra ke dalam sejumlah kelas, sehingga *output* akan menampilkan hasil identifikasi aksara berupa teks huruf, lalu setelah data teks aksara didapatkan, selanjutnya akan dilakukan proses *text to speech* untuk menampilkan *output* berupa suara, sehingga hasil akhirnya sistem dapat menampilan *output* berupa pengenalan aksara beserta cara pengucapannya.

*1) Akuisisi citra :* Akuisisi citra merupakan tahap pengambilan citra kedalam sistem. Berikut merupakan contoh citra yang sudah di load dengan nama file "i.jpg". seperti yang terlihat pada Gambar 5.



Gambar 5. *Load Image* i.jpg

*2) PreProcessing :Image pre-processing* dilakukan untuk mendapatkan citra karakter yang sebaik mungkin untuk proses penggantian pixel pembentuk karakter dan pembangunan ciri. *Image pre-processing* yang biasa digunakan untuk mendukung MDF diantaranya yaitu *resize*, *grayscale*, dan *thinning* Gambar 6 merupakan hasil dari proses *pre-processing*.

- *Resizing*

*Resizing* merupakan pengubahan ukuran *image* agar seluruh citra memiliki ukuran seragam[puspa]. Pengubahan ukuran piksel pada citra dengan skala 15x15 piksel pada MATLAB menggunakan fungsi sebagai berikut :

```
img2 = imresize(image,[15,15]);
```

- *Grayscaling*

*Grayscale* merupakan tahap prapemrosesan yang bertujuan untuk mendapatkan citra keabuan[RD], untuk proses *grayscale* pada MATLAB menggunakan fungsi sebagai berikut :

```
gray = rgb2gray(img2);
```

- *Thinning*

*Thinning* adalah operasi morphology yang digunakan untuk mengikis piksel latar depan sampai menjadi *pixel* yang tipis. Citra yang digunakan untuk proses *thinning* adalah citra biner. Citra biner ini yang kemudian akan diproses dalam proses *thinning[suci]*, untuk proses *thinning* pada MATLAB menggunakan fungsi sebagai berikut :

```
thin=bwmorph(~img3, 'thin', inf);
```


Gambar 6. proses *pre-processing*

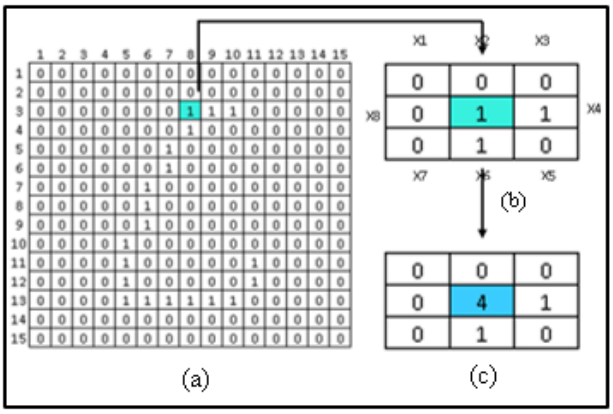
*3) Tahap Ekstraksi Ciri MDF :* Ekstraksi ciri adalah proses untuk mendapatkan ciri dari citra karakter. Ekstraksi ciri pada sistem yang dibangun ini adalah dengan menggunakan metode MDF. MDF merupakan tahap awal yang yang digunakan untuk identifikasi informasi pada sebuah citra yang akan di uji. Fitur atau ciri adalah karakteristik unik dari suatu objek. Tujuan dari ekstraksi ciri ini adalah mendapatkan karakteristik suatu karakter yang berguna untuk membedakan antara karakter yang satu dengan lainnya. Ciri yang baik adalah ciri yang memiliki daya pembeda tinggi, sehingga proses klasifikasi pada ciri karakter bisa mendapatkan akurasi yang baik. Pada subbab ini disimulasikan perhitungan MDF pada aksara sunda "I"dengan ukuran lebar dan panjang [15x15], 4 arah (kiri, kanan, atas, bawah), jumlah transisi 3, normalisasi ciri panjang 5, dan 2 nCiri (LT dan DT).

- *Direction Feature (*DF) adalah pencarian nilai *feature* berdasarkan arah dari sebuah *pixel.* Pada metode ini setiap *pixel foreground* (*pixel* yang memiliki nilai 1) memiliki label arah tersendiri yang terdiri dari 4 nilai arah. Arah yang digunakan pada pelabelan tersebut dengan menggunakan aturan seperti yang terlihat pada Tabel I dan Gambar 1. Gambar 7 merupakan citra hasil proses *thinning* dengan piksel bernilai 1.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Gambar 7. Pemberian nilai arah *pixel* bernilai 1

Adapun ilustrasi proses pemberian nilai arah menurut aturan pada piksel [8,3] untuk aksara sunda "i" seperti yang ditunjukan oleh `Gambar 8` adalah sebagai berikut :



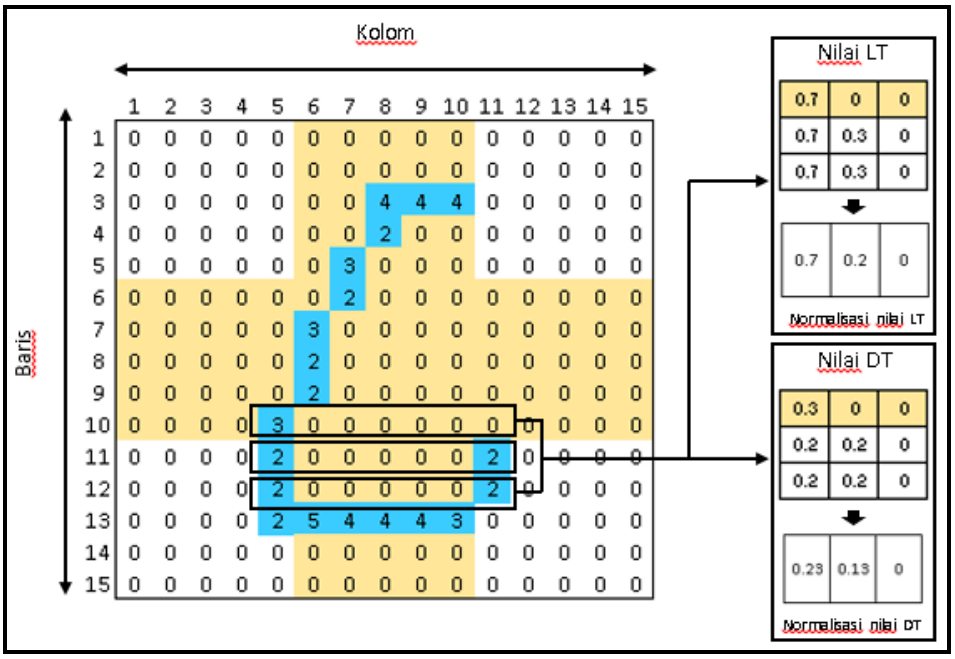

Gambar 8. Pemberian nilai arah *pixel* bernilai 1

Berdasarkan aturan yang telah ditetapkan karena pada saat pengecekan X pada *pixel foreground* [8,3] (Gambar 8(a)) sudah ditemukan X pertama yang merupakan *foreground* pada posisi X4 yang ditunjukan oleh Gambar 8 (b) maka nilai arah yang diberikan adalah 4 seperti yang terlihat pada Gambar 8 (c). Begitu seterusnya untuk melakukan pelabelan arah hingga semua *pixel foreground* terpenuhi seperti yang terlihat pada Gambar 9 .

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 4 | 4 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 2 | 5 | 4 | 4 | 4 | 3 | 0 | 0 | 0 | 0 | 0 |
| 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Gambar 9. Nilai keseluruhan hasil pemberian nilai arah

- *Trantition Feature (TF)* adalah proses untuk menghitung posisi transisi dan jumlah transisi. Setelah *pixel foreground* yang bernilai 1 diberi nilai sesuai *gesture* arah ketetanggaan, selanjutnya akan dihitung posisi dan jumlah transisi dari suatu piksel citra dengan mengggunakan rumus pencaian nilai LT dan DT. Gambar 10 merupakan contoh ilusrasi perhitungan nilai LT dan DT dari arah kiri ke kanan pada posisi [5,10], [5,11], [11,11], [5,12], dan [12,12].




Gambar 10. Perhitungan nilai LT dan DT

Pada Gambar 10 diketahui piksel [5,10] terdapat transisi, sehingga berlaku persamaan 1:

$$LTi = 1 - \left(\frac{xi}{Maxi}\right) = 1 - \left(\frac{5}{15}\right) = 0.7$$

Dimana $xi$ adalah indeks pixel yang dikaji dihitung dari awal pencarian dan $Maxi$ merupakan jumlah *pixel* maksimal dalam satu baris atau kolom segmen *pixel* citra. Dikarenakan pada penelitian ini menggunakan 3 transisi dan normalisasi matriks 5, maka jumlah transisi yang ditemukan kurang dari jumlah maksimal nilai transisi sisanya diberikan nilai 0. Selanjutnya setelah semua nilai transisi pada setiap baris didapatkan seperti yang ditunjukan oleh Gambar 10 nilai LT, maka dilakukan normalisasi dengan membagi ketiga nilai pada setiap 3 baris dari 1 transisi, sehingga didapat nilai rata-rata.

$$nLTi = \left(\frac{0.7+0.7+0.7}{3}\right) = 0.7$$

Dimana $xi$ adalah nilai piksel yang telah dikaji dan dihitung dari awal pencarian dan $MaxTransisi$ merupakan jumlah maksimal transisi dalam satu baris segmen *pixel* citra, dalam kasus ini karena menggunakan 3 transisi sehingga nilai pembaginya adalah 3.

Setelah nilai TF ditemukan dari masing-masing LT yang telah dicari, selanjutnya akan dilakukan perhitungan DT yang dilakukan pada setiap transisi yang terjadi pada TF. Nilai DT dari DF pada MDF diambil dari pembagian antara nilai DF dengan 10.

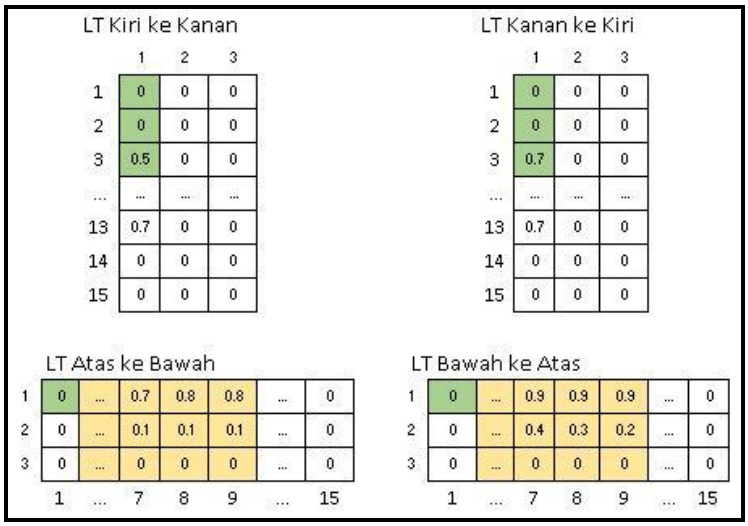
Pada Gambar 10 diketahui piksel [5,10] terdapat transisi, sehingga untuk nilai DT berlaku persamaan 3:

$$DTi = \left(\frac{DFi}{Maxi}\right) = \left(\frac{3}{10}\right) = 0.3$$

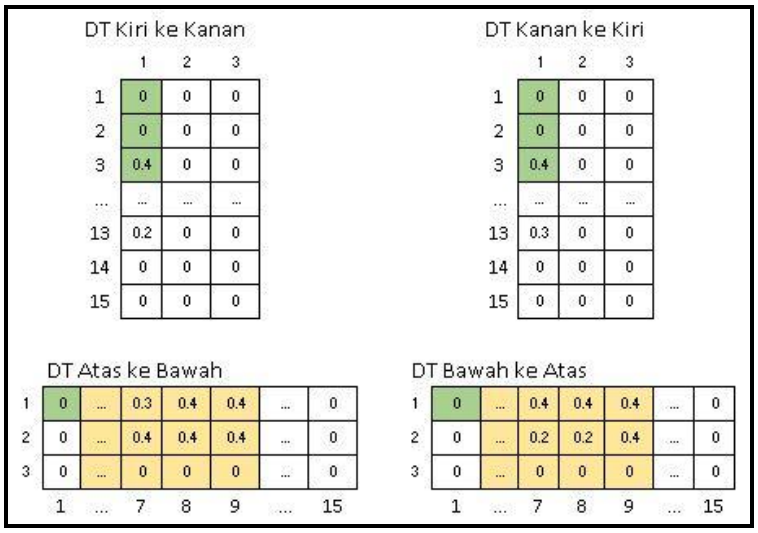
Selanjutnya setelah semua nilai DT pada setiap transisi didapatkan seperti yang ditunjukan oleh Gambar 10 nilai DT, maka dilakukan normalisasi dengan membagi ketiga nilai pada setiap 3 baris dari 1 transisi, sehingga didapat nilai rata-rata.

$$nLTi = \left(\frac{0.3+0.2+0.2}{3}\right) = 0.23$$

Berikut merupakan contoh nilai keseluruhan LT dan DT yang didapatkan dari perhitungan nilai transisi, seperti ditunjukan oleh Gambar 11a untuk nilai LT dan Gambar 11b untuk nilai DT.

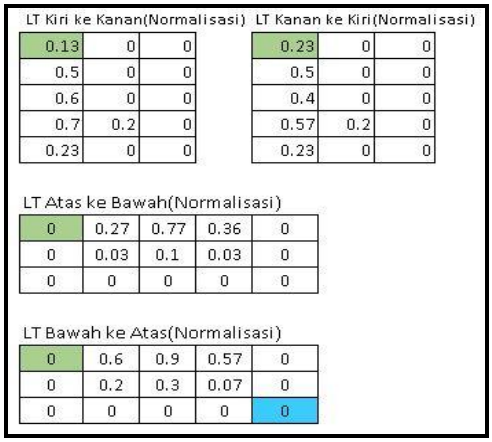



Gambar 11a. Nilai LT

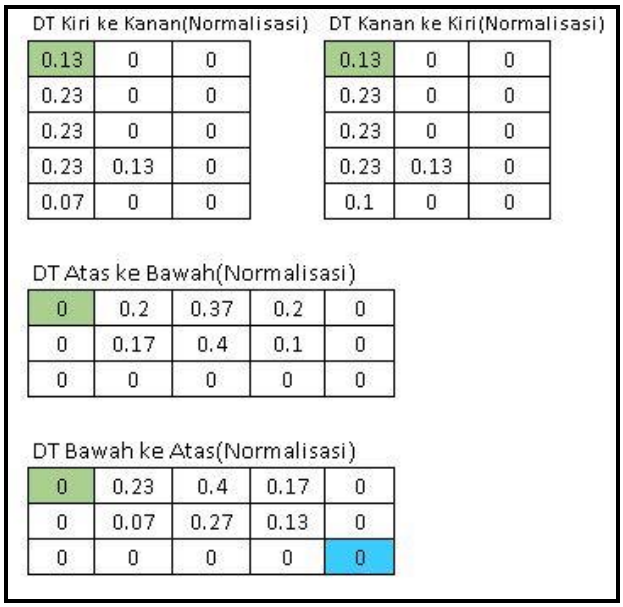



Gambar 11b. Nilai DT

Selanjutnya setelah semua nilai transisi didapatkan, maka dilakukan normalisasi dengan membagi ketiga nilai pada setiap 3 baris dari 1 transisi, sehingga didapatkan jumlah ukuran matriks sebanyak 5. Berikut adalah hasil perhitungan keseluruhan nilai normalisasi LT yang ditunjukan oleh Gambar 12a dan hasil perhitungan keseluruhan nilai normalisasi DT dapat dilihat pada Gambar 12b.




Gambar 12a. Nilai keseluruhan LT



DT Kiri ke Kanan(Normalisasi)

| | | |
|---|---|---|
| 0.13 | 0 | 0 |
| 0.23 | 0 | 0 |
| 0.23 | 0 | 0 |
| 0.23 | 0.13 | 0 |
| 0.07 | 0 | 0 |

DT Kanan ke Kiri(Normalisasi)

| | | |
|---|---|---|
| 0.13 | 0 | 0 |
| 0.23 | 0 | 0 |
| 0.23 | 0 | 0 |
| 0.23 | 0.13 | 0 |
| 0.1 | 0 | 0 |

DT Atas ke Bawah(Normalisasi)

| | | | | |
|---|---|---|---|---|
| 0 | 0.2 | 0.37 | 0.2 | 0 |
| 0 | 0.17 | 0.4 | 0.1 | 0 |
| 0 | 0 | 0 | 0 | 0 |

DT Bawah ke Atas(Normalisasi)

| | | | | |
|---|---|---|---|---|
| 0 | 0.23 | 0.4 | 0.17 | 0 |
| 0 | 0.07 | 0.27 | 0.13 | 0 |
| 0 | 0 | 0 | 0 | 0 |

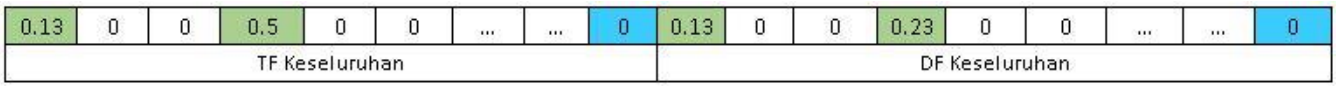
Gambar 12b. Nilai keseluruhan DT

Setelah didapatkan nilai normalisasi LT dan DT dari ke empat arah, maka nilai normalisasi tersebut disatukan dalam satu baris vektor. Maka vektor inilah yang akan menjadi vektor ciri dari citra karakter tersebut. Dengan adanya perhitungan matriks LT dan DT yang dihitung dari keempat arah pencarian, maka dilakukan normalisai vektor ciri dengan menggunakan persamaan 4, sehingga akan dihasilkan vektor sejumlah :

$$vektor\ ciri = 2\ x\ 3\ x\ 4\ x\ 5 = 120$$

Dimana : 2 adalah LT dan DT
3 adalah jumlah transisi yang dipakai
4 adalah jumlah arah pencarian
5 adalah jumlah ukuran normalisasi matriks

Berikut merupakan hasil penggabungan nilai keseluruhan nilai normalisasi LT dan DT yang ditunjukan oleh Gambar 12a dan Gambar 12b, sehingga hasil akhir vektor ciri dari nilai MDF pembentuk aksara sunda "I" memiliki panjang 120 vektor ciri seperti yang terihat pada Gambar 13 untuk aksara sunda "I".




Gambar 13. Nilai vektor ciri LT dan DT

*4) Tahap Klasifikasi :* Setelah nilai ciri didapatkan dari tahap ekstraksi ciri menggunakan metode MDF maka proses selanjutnya yang dilakukan adalah tahap klasifikasi dengan menggunakan metode LVQ.Untuk penerapannya yaitu dimisalkan didapat vektor dari data uji (0.289, 0, 0, 0.422, 0, …, 0, 0, 0.367, 0, 0). Data latih berupa bobot ($W_1$ dan $W_2$) dan dimisalkan terdapat dua kelas target ($T_1$ dan $T_2$).

- $W_1$ untuk kelas $T_1$ yaitu (0.333, 0, 0, 0.4, 0, …, 0, 0, 0.367, 0, 0) dan
- $W_2$ untuk kelas $T_2$ yaitu (0.267, 0.2, 0, 0.244, 0.067, …, 0.1, 0, 0.133, 0.167, 0).

Penghitungan jarak *Euclidean* antara vektor input dan vektor bobot dilakukan dengan menggunakan persamaan 5. Maka dengan nilai bobot $W_1$ dan $W_2$, perhitungan untuk mencari nilai bobot terkecil yaitu :

- Jarak pada bobot ke-1 =

$$= \sqrt{\begin{matrix}(0.289-0.333)^2+(0-0)^2+(0-0)^2+(0.422-0.4)^2+(0-0)^2+\dots \\ +(0-0)^2+(0-0)^2+(0.376-0.376)^2+(0-0)^2+(0-0)^2\end{matrix}}$$

$= 0.395$

- Jarak pada bobot ke-2 =

$$= \sqrt{\begin{matrix}(0.289-0.267)^2+(0-0.2)^2+(0-0)^2+(0.422-0.24)^2+(0-0.067)^2+\dots \\ +(0-0.1)^2+(0-0)^2+(0.376-0.133)^2+(0-0.167)^2+(0-0)^2\end{matrix}}$$

$= 2.199$

Dari hasil perhitungan jarak diatas, nilai jarak paling kecil yaitu nilai jarak pada bobot ke-1, dengan nilai 0.395. Sehingga dapat disimpulkan bahwa input vektor nilai MDF yang diuji tersebut termasuk ke dalam Kelas pertama ($T_1$).

*5)Tahap Teks to Speech :* Proses selanjutnya untuk dapat memainkan suara contoh pengucapan aksara dilakukan teknik pemanggilan suara menggunakan algoritma KMP. Penerapannya adalah dengan spesifikasi sebagai berikut:

1. *Library file* :

| |
|---|
| //a// //ba// //ca// //da// //e// //é// //eu// //ga// //ha// //i// //ja// //ka// //la// //ma// //na// //nga// //nya// //o// //pa// //ra// //sa// //ta// //u// //wa// //ya// |

2. *File* suara yang ada:

TABEL III

*FILE* SUARA

| No. | *File* Suara |
|---|---|
| 1 | a.wav |
| 2 | ba.wav |
| ... | ….. |
| 24 | wa.wav |
| 25 | ya.wav |

3. Pertama melakukan sintesis dari teks “i” ke dalam suara.

Teks: a

Lalu program akan melakukan *searching* teks “i”.

4. *Search*:

*Pattern*: i

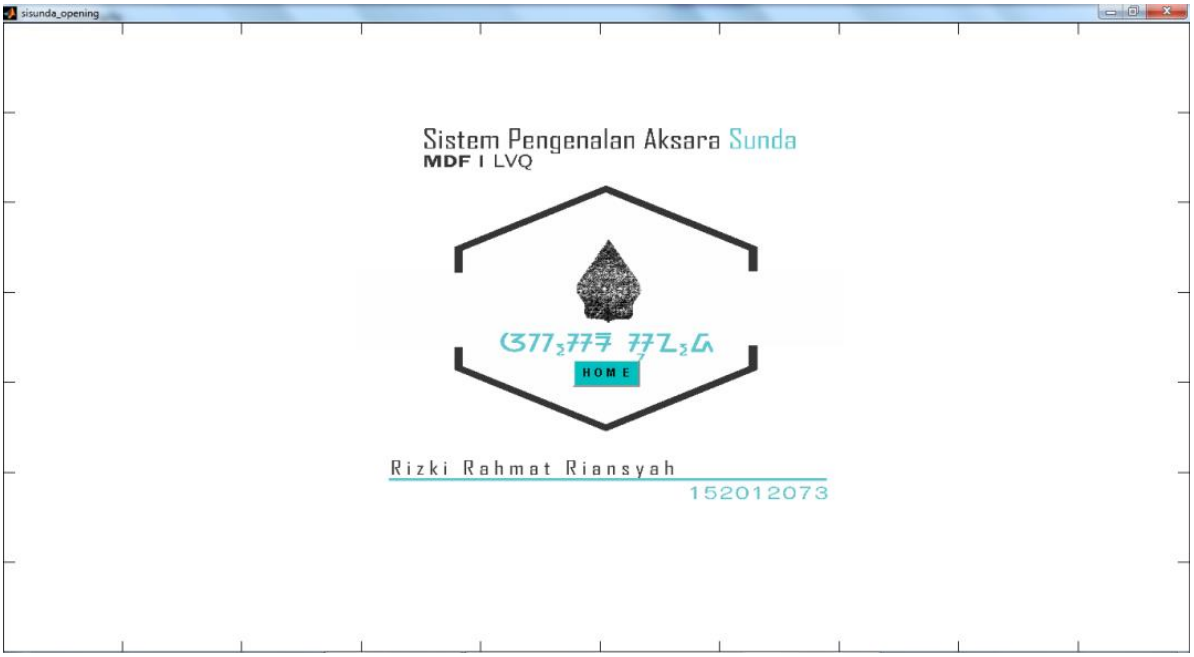
*String* yang dicocokkan: isi dari *library file*

*Pattern* ditemukan dalam *string* yang dicocokkan, maka program akan memainkan file “i.wav”.

## V. IMPLEMENTASI

### *A. Tampilan Aplikasi*

Terdapat beberapa buah halaman pada aplikasi yang dibuat. Beberapa diantaranya adalah halaman pembuka aplikasi (Gambar 14) yang berisi judul dan tombol untuk masuk ke halaman utama aplikasi. Pada halaman selanjutnya adalah halaman pengujian untuk melakukan pencocokan terhadap citra aksara sunda seperti yang ditunjukan oleh Gambar 15.




Gambar 14. Halaman pembuka

Pada halaman pengujian (Gambar 15) terdapat akses untuk mengambil citra yang akan diuji dengan menekan tombol Image yang akan dilakukan proses *preprocessing*, lalu citra akan diproses untuk mendapatkan nilai cirinya agar dapat dilakukan pencocokan terhadap data latih, selanjutnya setelah citra yang dipilih muncul pada halaman pengujian, *user* dapat menekan tombol Matching untuk mengetahui *output* dari proses pencocokan aksara sunda yang terdapat pada kolom keterangan hasil pengujian kecocokan aksara sunda dengan nama dan contoh pengucapannya, serta terdapat tombol untuk mendengarkan ulang suara yang dihasilkan. Sebagai contoh pada Gambar 15 telah diketahui suatu pengujian citra aksara dengan *output* aksara “I” dari citra aksara uji “I”.


Gambar 15. Halaman pengujian

*B. Pengujian*

Pengujian dilakukan dengan mencocokkan 25 citra aksara sunda kaganya ( ba, ca, da, ga, ha, ja, ka, la, ma, na, nga, nya, pa, ra, sa, ta, wa, ya ) dan swara (a, i, u, e, é, eu, o ) pada data latih. Teknik pengambilan data citra aksara sunda kaganga dan swara tersebut dilakukan dengan menggunakan *scanner*, kamera, dan *screenshot* pada teks digital serta dengan ukuran aksara 20pt, 72pt, dan 150 pt, ukuran tersebut diambil secara *random* dengan memperhatikan aspek ukuran ideal yang biasa digunakan dalam sebuah teks bacaan sebagai pengujian untuk mengetahui seberapa baik sistem yang dibangun dalam mengenali masing-masing karakter aksara sunda dari beberapa ukuran. Berikut merupakan hasil pengujian yang telah dilakukan dengan melakukan pencocokan dengan 300 citra aksara sunda antara citra uji dengan citra latih yang terdiri dari 25 citra aksara sunda yang di *screenshots* dengan ukuran aksara 20pt hingga 25 citra aksara sunda yang dipotret dengan ukuran aksara 150pt, seperti yang ditunjukan oleh Tabel 4 dan Tabel 5 .

Tabel 4 merupakan beberapa contoh pengujian citra aksara yang dapat dikenali dengan benar dan citra aksara yang tidak dapat dikenali dengan benar sehingga menampilkan *output* teks yang salah, sedangkan Tabel 5 merupakan hasil persentase akurasi keseluruhan pengujian citra aksara.

TABEL IV

CONTOH PENGUJIAN AKSARA SUNDA

| No | Teknik Pengujian Aksara | Ukuran aksara | Bentuk Aksara Uji | Nama Aksara | Hasil Pengujian | Bentuk Aksara Dikenali | Nama Aksara |
|---|---|---|---|---|---|---|---|
| 1 | Citra aksara yang di *screenshots* | 20pt |  | ba | Dikenali |  | ba |
| 2 | | 72pt |  | a | Tidak Dikenali |  | é |
| 3 | Citra aksara yang di pindai | 72pt |  | ca | Dikenali |  | ca |
| 4 | | 150pt |  | ga | Tidak Dikenali |  | u |
| 5 | Citra aksara yang di potret dengan kamera *Handphone*(13MP) | 20pt |  | da | Dikenali |  | da |
| 6 | | 72pt |  | ya | Tidak Dikenali |  | la |
| 7 | Citra aksara yang di potret dengan *Digicam*(16MP) | 72pt |  | i | Dikenali |  | i |
| 8 | | 150pt |  | ka | Tidak Dikenali |  | sa |

TABEL V

HASIL PENGUJIAN AKSARA SUNDA

| No | Teknik Pengujian Aksara | Ukuran aksara | Persentase Hasil Pengujian |
|---|---|---|---|
| 1 | 25 Citra aksara yang di *screenshots* | 20pt | 84% |
| 2 | | 72pt | 84% |
| 3 | | 150pt | 84% |
| 4 | 25 Citra aksara yang di pindai | 20pt | 84% |
| 5 | | 72pt | 84% |
| 6 | | 150pt | 88% |
| 7 | 25 Citra aksara yang di potret dengan kamera *Handphone*(13MP) | 20pt | 44% |
| 8 | | 72pt | 64% |
| 9 | | 150pt | 76% |
| 10 | 25 Citra aksara yang di potret dengan *Digicam*(16MP) | 20pt | 84% |
| 11 | | 72pt | 84% |
| 12 | | 150pt | 88% |

Berdasarkan pengujian yang dilakukan dengan 300 citra aksara sunda mendapatkan tingkat akurasi kesesuaian citra aksara dengan nama dan pengucapannya dalam persen yaitu dengan menggunakan persamaan 7 :

$$Hasil = \frac{Hasil\ yang\ sesuai}{Total\ citra\ yang\ diuji} \times 100 \quad \text{.....(7)}$$

Sehingga nilai akurasi keberhasilan pengujian untuk 300 citra aksara sunda yang diuji yaitu :

$$Hasil = \frac{236}{300} \times 100 = 78{,}67\ \%$$

Berdasarkan hasil pengujian tersebut didapatkan akurasi keberhasilan sebesar 78.67% dari 300 sampel data uji, hasil akurasi tersebut diperoleh dari nilai kedekatan jarak minimum antara *vector input* dengan vektor bobot seperti yang ditunjukan oleh Tabel 5. sehingga dapat menghasilkan *output* citra aksara sunda beserta kesesuaian contoh pengucapannya. Keberhasilan pencocokan citra uji tersebut berdasarkan hasil pengujian sistem sudah cukup baik terbukti dari beberapa penelitian yang telah dilakukan menghasilkan tingkat akurasi dibawah 70%, dimana pada beberapa penelitian tersebut hanya melakukan pencocokan pada satu ukuran karakter uji dan satu teknik pengambilan gambar saja, sedangkan pada penelitian ini menggunakan beberapa ukuran citra aksara dan beberapa teknik pengambilan citra aksara.

Hasil pengujian terbaik terdapat pada pengujian dengan citra aksara yang di pindai dan di potret dengan menggunakan kamera *digicam* sebesar 16MP dengan ukuran aksara 150pt yaitu sebesar 88%, sedangkan hasil pengujian terendah terdapat pada pengujian dengan citra aksara yang di potret dengan menggunakan kamera *handphone* sebesar 13MP dengan ukuran aksara 20pt yaitu sebesar 44%. Hal ini dapat dikarenakan jumlah data pelatihan, kualitas dan struktur citra aksara yang diambil sebagai data uji yang rendah sehingga pada saat proses identifikasi nilai ciri sebagai nilai input dari citra uji aksara menghasilkan nilai jarak *euclidean* minimum yang terlalu dekat dengan nilai bobot aksara lain, sehingga hasil pengujian mengalami ketidak sesuaian antara bentuk aksara dengan nama aksara yang seharusnya, dikarenakan nilai *input* aksara tersebut tidak terklasifikasi dengan benar, sehingga nilai ciri aksara uji mengacu kepada nilai ciri kelas lain yang bukan merupakan kelas citra uji yang seharusnya, oleh karena itu *output speech* yang dihasilkanpun mengalami ketidak sesuaian, karena pada sistem yang dibangun *speech* bergantung pada kelas LVQ.

## VI. PENUTUP

### A. *Kesimpulan*

Berdasarkan penelitian yang telah dilakukan menggunakan metode MDF yang digunakan untuk proses ekstraksi ciri, pada dasarnya metode MDF mampu mengenali struktur pembentuk dari citra yang dipilih dengan nilai tingkat pembeda yang tinggi yang diperoleh dari hasil transformasi vektor arah dan transisi dari suatu pola citra biner menjadi sebuah vektor yang mewakili ciri dari citra tersebut. Namun dalam tahap pencocokan aksara, metode LVQ dalam prosesnya melakukan pembelajaran secara kompetitif terawasi untuk mengklasifikasi vector input hanya mampu mengenali 236 aksara dari 300 aksara citra uji.

Berdasarkan hasil penelitian serta pengujian sistem seperti yang ditunjukan pada Tabel 5 yang telah dilakukan, maka dapat disimpulkan bahwa sistem dapat digunakan untuk mengenali cira aksara sunda dengan total rata-rata tingkat akurasi keberhasilan dalam pengenalan aksara sunda

beserta kesesuaian contoh pengucapannya adalah sebesar 78,67%. Tingkat akurasi tersebut dipengaruhi oleh jumlah data pelatihan, ukuran aksara dan teknik pengambilan citra aksara.

### *B. Saran*

Berdasarkan hasil pengujian yang telah dilakukan terdapat beberapa saran agar dapat diperbaiki dan dikembangkan di masa yang akan datang. Beberapa saran tersebut adalah sebagai berikut:

1. Sistem pengenalan dapat mengenali kata bahkan kalimat, serta sistem mampu mengenali berbagai macam citra hingga tulisan tangan.
2. Menambahkan atau mengubah metode pengolahan citra dengan metode lain agar proses identifikasi dapat lebih akurat.
3. Sistem TTS diharapkan dapat mengeluarkan bunyi dengan intonasi yang lebih baik, serta sistem TTS dapat diimplementasikan tidak hanya dengan teknik pencarian string, namun dengan menggunakan algoritma TTS.

## DAFTAR PUSTAKA

[1] Abi Mahan Zaky, "Implementasi Algoritma Knuth Morris Pratt Pada Perancangan *Game* Hanacaraka," Semarang, Skripsi Teknik Elektro Universitas Negeri Semarang, 2015.

[2] Dannis Muhammad Mangan, "Pembuatan *Tect-to-Speech* Sederhana Menggunakan Algoritma KMP untuk Pencocokan String," *Makalah Teknik Informatika ITB*, 2009.

[3] Dea Delia, Dr. Ir. Bambang Hidayat, DEA, Nur Andini, S.T., MT., "Perancangan Pengenal Kata Dalam Aksara Sunda Menggunakan Metode Deteksi Tepi dan LVQ Berbasis Pengolahan Citra Pada Android," Bandung, Jurnal Teknik Telekomunikasi Universitas Telkom, 2015.

[4] Erik Farhan Malik, Bedy Purnama, Mahmud Dwi Sulistiyo, "Analisis dan Implementasi *Optical Character Recognition* (OCR) Menggunakan Jaringan Syaraf Tiruan (JST) untuk Pengenalan Aksara Sunda Baku," Bandung, Jurnal Informatika Institut Teknologi Telkom, 2012.

[5] Fachrul Kurniawan, Hani Nurhayati, "Simulasi Pengenalan Tulisan Mengunakan LVQ (Learning Vector Quantization)," *Jurnal Teknik Informatika UIN*.

[6] Fitri Damayanti, Wahyudi Setiawan, "Pengenalan Tanda Tangan Dengan Metode *Modified Direction Feature* (MDF) dan *Euclidean Distance*," Prosiding Conference on Smart-Green Technology in Electrical and Information Systems, Bali, 2013.

[7] Gilang Rachman Perdana, Deni Saepudin, Adiwijaya, "Analisis dan Implementasi Optical Character Recognition Menggunakan Metode Modified Direction Feature dan Least Squares Support Vector Machine," Bandung , Institut Teknologi Telkom.

[8] Hendro Triokta Brianto, "Konversi Alfabet Latin Ke Aksara Sunda", Bandung, Makalah ITB, 2014.

[9] Jasman Pardede, Youllia Indrawaty Nurhasanah, Rd. Rakha Agung Trimanda, "Perbandingan Metode *Diphone Concatenation* dan Algoritma *Sonic* pada *Tect-to-Speech*," *Jurnal Teknik Informatika ITENAS*, 2016.

[10] Joko Aris Pramono, "Rancang Bangun Aplikasi Text To Speech Sebagai Alat Bantu Pembelajaran Bahasa Inggris". Yogyakarta: STMIK AMIKOM, 2013.

[11] Maulana Nur Muhamad, "Implementasi Algoritma Chain Code dan LVQ pada Pengenalan Huruf Hijaiyah," Bandung:ITENAS, 2016.

[12] Michael Blumenstein, XinYu Liu, Brijesh V, *An investigation of the modified direction feature for cursive character recognition, sciencedirect Patern Recognition 40*, pp 376-388, 2007.

[13] Murni, Aniati, Pengantar Pengolahan Citra. Jakarta: Penerbit PT Elex Media komputindo, 1992.

[14] Pemerintah Provinsi Jawa Barat, Dinas Pendidikan Provinsi Jawa Barat, Direktori Aksara Sunda untuk Unicode, Bandung, 2008.

[15] RD. Kusumanto, Alan Novi Tompunu, "Pengolahan Citra Digital Untuk Mendeteksi Obyek Menggunakan Pengolahan Warna Model Normalisasi RGB," Palembang, Semnas Teknologi Informasi & Komunikasi, 2011.

[16] Rinaldi Munir, Pengolahan Citra Digital. Jakarta : Penerbit Informatika.

[17] Suci Indah Syahputri, "Analisis dan Perancangan Perangkat Lunak *Image Thinning* dengan Metode Zhang Suen," Medan, Skripsi Universitas Sumatera Utara Program Studi S1 Ilmu Komputer, 2011.

[18] Siang, Jong Jek, Jaringan Syaraf Tiruan dan Pemrogamannya Menggunakan MATLAB, Yogyakarta : Penerbit Andi, 2014.

[19] Tjokro Agung BW, I Gede Rudy Hermanto, Retno Novi D, "Pengenalan Huruf Bali Menggunakan Metode Modified Direction Feature (MDF) dan Learning Vektor Quantization (LVQ)," Bandung , Institut Teknologi Telkom, 2009.