ISSN : 2407 - 3911

OPTICAL CHARACTER RECOGNITION JEPANG MENGGUNAKAN MATRIKS POPULASI PIKSEL DAN *L1-METRIC*

Ade Setiawan[1)], Kristien Margi Suryaningrum[2)]
Technology and Design Departmen, Informatics Engineering
Universitas Bunda Mulia
Jalan Lodan Raya No.2, Ancol, Jakarta Utara
geowan95@gmail.com[1)], ksuryaningrum@bundamulia.ac.id[2)]

ABSTRAK

Aksara *Hiragana* dan *Katakana* merupakan bahasa yang berasal dari negara Jepang. Bahasa Jepang juga telah menyebar di Indonesia khususnya dalam pembelajaran. Namun dikarenakan bahasa Jepang bukanlah bahasa dari daerah Indonesia dan bukan bahasa Internasional maka bahasa ini sulit dipelajari dan dibaca bagi seorang pemula yang ingin mempelajari bahasa tersebut. Oleh karena itu, diperlukan sebuah sistem yang dapat membaca aksara Jepang. Penelitian ini akan difokuskan pada perancangan aplikasi pengenalan karakter optik aksara Jepang dengan menggunakan fitur matriks populasi piksel dan *l1-metric distance* dalam melakukan pengenalan pada citra aksara Jepang dimana fitur matriks populasi piksel digunakan untuk mendapatkan fitur dari karakter dan *l1-metric distance* untuk pengukuran jarak. Serta dalam membangun aplikasi digunakan bahasa pemograman *java*. Proses dalam penelitian ini terdiri dari 4 tahap *yaitu pre-processing*, segmentasi, ekstrasi fitur, dan pengukuran jarak. *Pre-processing* dilakukan dengan proses binerisasi. Segmentasi dilakukan dengan segmentasi karakter dalam sebuah kata. Ekstrasi fitur dilakukan untuk mendapatkan fitur dari aksara. Pengukuran jarak dilakukan untuk menghitung selisih antara data yang diuji dengan data yang terdapat dalam basis data lalu membandingkan dimana jarak yang terkecil merupakan data sampel yang mendekati data uji.

Hasil pengujian yang dilakukan dengan fitur matriks populasi piksel dan *l1-metric distance* menunjukan bahwa 82,61% aksara jepang dengan jenis *font* yang berbeda dengan data sampel berhasil dikenali. Namun masih 17,39% masih terdapat kegagalan dalam mengenali aksara tersebut. Kegagalan ini dikarenakan proses binerisasi dan segmentasi yang kurang baik dimana proses binerisasi kadang menghilangkan piksel-piksel yang seharusnya tidak hilang dan segmentasi kurang mampu memisahkan per karakter.

Kata kunci :

Hiragana, *Katakana*, OCR, Matriks Populasi Piksel, *L1-Metric*.

ABSTRACT

Hiragana and Katakana is a language that comes from Japan. Japanese language has spread in Indonesia especially in learning. But because of Japanese language is not a language from Indonesia and not International language then this language is difficult to learn and read for a beginner who wants to learn the language. Therefore, it takes a system that can read Japanese script. This research will be focused on design application of optical character recognition of Japanese scriptby using pixel population matrix and l1-metric distance in the introduction of Japanese image where pixel population matrix is used to obtain feature of character and l1-metric distance for distance measurement. And in building applications used java programming language. The process in this research consists of 4 stages of pre-processing, segmentation, feature extraction, and distance measurement. Pre-processing is done by binary process. Segmentation is done by segmenting characters in a word. Feature extraction is done to get the feature of the script. Distance measurements are performed to calculate the difference between the data tested and the data contained in the database and then compare where the smallest distance is the sample data close to the test data.

The test results were performed with the pixel population matrix and l1-metric distance indicates that 82.61% japanese characters with different font types with sample data are identified. But still 17.39% there are still failures in recognizing the script. This failure is due to the poor binary and segmentation process where the binary process sometimes removes the pixels that should not be lost and the segmentation is less able to separate per character.

Keywords :

Hiragana, Katakana, OCR, Pixel Population Matrix, L1-Metric.

I. PENDAHULUAN

Banyaknya pengemar anime, *cosplay*, dan film Jepang di Indonesia telah membuat sejumlah orang tertarik dalam mempelajari bahasa Jepang. Pada dasarnya bahasa Jepang bukanlah bahasa yang berasal dari Negara Indonesia melainkan bahasa yang berasal dari Negara Jepang dan bahasa Jepang juga bukan merupakan bahasa international. Dikarenakan bahasa Jepang ini bukan bahasa internasional dan bukan bahasa dari negara Indonesia maka kebanyakan orang menjadi kesulitan dalam mempelajari bahasa Jepang.

Tidak sama seperti bahasa Indonesia yang tidak memiliki aksara, bahasa Jepang mempunyai aksara yaitu aksara *Hiragana* dan *Katakana* dimana aksara *Hiragana* untuk kata-kata sehari-hari dan aksara *Katakana* untuk kata-kata serapan. Dikarenakan kesulitan-kesulitan tersebut maka pemula yang ingin mempelajari bahasa Jepang menjadi kesulitan dalam mempelajari bahasa tersebut. Pelajaran disekolah pun terkadang mengajarkan bahasa Jepang sebagai salah satu pembelajaran bahasa maupun ekstrakulikuler bahasa Jepang di sekolah. Pembelajaran bahasa Jepang juga terdapat di berbagai kursus bahasa Jepang di Indonesia

Melihat bagaimana bahasa jepang telah menyebar di Indonesia maka peneliti akan merancang dan membuat sebuah aplikasi *Optical Character Recognition* terhadap aksara atau huruf dari *Hiragana* dan *Katakana* agar dapat membantu seseorang dalam mempelajari bahasa Jepang. Aplikasi *Optical Character Recognition* merupakan aplikasi yang dirancang dengan menerapkan *computer vision* untuk mengenali karakter-karakter berbentuk gambar menjadi sebuah informasi yang sesuai dan berguna. Aplikasi pengenal huruf *Hiragana* dan *Katakana* dibuat dengan mengimplementasikan fitur matriks populasi piksel agar menghasilkan informasi kanji sesuai dengan aksara *Hiragana* dan *Katakana*.

Berdasarkan latar belakang tersebut maka dapat dirumuskan masalah sebagai berikut yaitu bagaimana fitur matriks populasi piksel dapat diimplementasikan pada aplikasi OCR aksara Jepang, bagaimana *L1-Metric Distance* dapat diimplementasikan pada aplikasi OCR aksara Jepang, bagaimana fitur matriks populasi piksel dan *l1-metric* menghasilkan informasi huruf *kanji* dari aksara *Hiragana* dan *Katakana*?

Berdasarkan rumusan masalah tersebut, maka tujuan penelitian yaitu sebagai berikut yaitu untuk membuktikan fitur matriks populasi piksel dapat diterapkan pada OCR aksara Jepang, untuk membuktikan *l1-metric* dapat diterapkan pada OCR aksara Jepang, dan u ntuk mendapatkan informasi dari fitur matriks populasi piksel dan *l1-metric*

Manfaat dari penelitian ini adalah untuk membantu pelajar disekolah dan lembaga kursus bahasa Jepang dalam mempelajari bahasa Jepang sehingga dapat mengenali aksara *Hiragana* dan *Katakana* dan mengetahui huruf kanji dari aksara tersebut.

II. LANDASAN TEORI

II.1 Aksara *Hiragana*

Hiragana adalah huruf bahasa Jepang asli yang dibuat oleh orang Jepang. Huruf ini mempunyai fungsi sebagai kata-kata asli bahasa Jepang yang bukan kata serapan. (Haryanto, Mahardityawarman, Kusnaedi, & Priyanggodo, 2016)

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

Gambar 1. Aksara *Hiragana*

II.2 Aksara *Katakana*

Huruf *Katakana* biasa dipakai untuk menulis kata serapan dari bahasa asing. Sebagaimana alfabet, huruf *Katakana* dan *Hiragana* hanya mewakili satu bunyi tanpa arti. Walaupun kalimat dalam bahasa Jepang biasa terdiri dari *Hiragana*, *Katakana* dan *Kanji*, tetapi bisa juga cuma ditulis dalam *Hiragana* dan *Katakana*. (Haryanto, Mahardityawarman, Kusnaedi, & Priyanggodo, 2016).

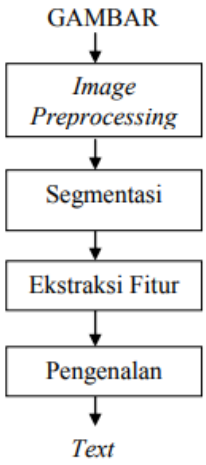
かたかな Katakana									
Seion	ア	a	イ	i	ウ	u	エ	e	オ	o
	カ	ka	キ	ki	ク	ku	ケ	ke	コ	ko
	サ	sa	シ	shi	ス	su	セ	se	ソ	so
	タ	ta	チ	chi	ツ	tsu	テ	te	ト	to
	ナ	na	ニ	ni	ヌ	nu	ネ	ne	ノ	no
	ハ	ha	ヒ	hi	フ	fu	ヘ	he	ホ	ho
	マ	ma	ミ	mi	ム	mu	メ	me	モ	mo
	ヤ	ya			ユ	yu			ヨ	yo
	ラ	ra	リ	ri	ル	ru	レ	re	ロ	ro
	ワ	wa							ヲ	wo
	ン	n								

Gambar 2. Aksara *Katakana*

II.3 *Optical Character Recognition*

Optical Character Recognition (OCR) adalah sebuah aplikasi komputer yang digunakan untuk mengidentifikasi citra huruf maupun angka untuk dikonversi ke dalam bentuk *file* tulisan. Sistem pengenal huruf ini dapat meningkatkan fleksibilitas atau kemampuan dan kecerdasan sistem komputer. Sistem pengenal huruf yang cerdas sangat membantu usaha besar-besaran yang saat ini dilakukan banyak pihak yakni usaha digitalisasi informasi dan pengetahuan, misalnya dalam pembuatan koleksi pustaka digital, koleksi sastra kuno digital, dan lain-lain. (A., Erik, & Hakim, 2014).

Secara garis besar proses OCR dapat dijelaskan pada Gambar 3.



Gambar 3. Proses OCR

Didalam OCR, gambar yang berisi karakter yang ingin dikenali dilakukan *preprocessing*. *Preprocessing* adalah proses menghilangkan konten-konten yang tidak diinginkan seperti *noise* dan juga untuk memperbaiki kualitas gambar agar lebih mudah dikenali. Dalam tahap pertama *preprocessing* dilakukan *grayscalling*. *Grayscalling* adalah mengubah gambar yang berwarna menjadi gambar yang hanya memiliki derajat keabuan saja. Selanjutnya dalam tahap kedua dilakukan noise filtering, yaitu proses mengurangi atau mereduksi noise yang ada pada gambar. *Noise* yang terlalu banyak dapat mengurangi keakuratan dalam pengenalan karakter. Dan tahap terakhir dari preprocessing yaitu *thresholding*. *Thresholding* memisahkan konten yang akan dikenali dengan background dengan mengubah gambar menjadi hitam putih. Dengan berakhirnya tahap *thresholding* maka tahap *preprocessing* selesai dilakukan.

Tahap selanjutnya adalah segmentasi. Segmentasi melakukan pemisahan karakter yang berarea besar menjadi area yang lebih kecil seperti suatu kalimat menjadi kata-kata, dan kata menjadi karakter. Setelah dilakukan segmentasi maka dilakukan normalisasi. Normalisasi mengubah karakter karakter hasil segmentasi menjadi suatu karakter yang memiliki karakterisitik yang telah ditentukan, seperti dimensi *region* dan ketebalan karakter. Setelah proses normalisasi dilakukan maka selanjutnya dilanjutkan pada tahap ekstraksi fitur. Ekstraksi fitur dilakukan untuk mendapatkan karakteristik khas yang dimiliki oleh tiap-tiap karakter. Dan tahap akhir dalam proses OCR ini adalah *recognition*, dimana dilakukannya

pengenalan dengan cara membandingkan ciri-ciri fitur yang ingin dikenali dengan data yang telah tersimpan sebelumnya sesuai dengan algoritma pengenalan yang dipakai. Hasil perbandingan yang miriplah yang kemudian keluar menjadi suatu hasil pengenalan berupa teks.

II.4 Binerisasi

Binerisasi merupakan proses untuk mengubah suatu citra menjadi citra yang hanya terdiri dari warna hitam dan putih. (Septiarini, 2012).

II.5 Segmentasi

Segmentasi merupakan suatu proses pemisahan objek yang satu dengan objek yang lain dalam suatu citra, berdasarkan sifat-sifat tertentu dari citra yang dapat dijadikan sebagai pembeda. (Septiarini, 2012).

II.6 Ekstrasi Fitur

Proses ekstrasi fitur terdiri dari proses pencarian batas atas, batas kiri, batas kanan, dan batas atas. Proses pencarian batas dilakukan dengan *scanline* secara horizontal dan secara vertikal. Untuk mendapatkan batas atas objek citra dilakukan *scanline* vertikal baris perbaris, mulai dari atas citra sampai ditemukan baris pertama yang berisi piksel hitam. Untuk mendapatkan batas bawah objek citra dilakukan *scanline* vertikal baris perbaris mulai dari bawah citra sampai ditemukan baris pertama yang berisi piksel hitam.

Setelah objek citra dibatasi dan diperoleh, selanjutnya adalah menggunakan fitur populasi matriks 5 x 5 bagian atau 25 bagian sel. Setiap sel akan berisi nilai yang merupakan populasi piksel dari setiap area citra yang merupakan luas dari 1/25 luas citra dalam hitungan piksel. (Mulyana, 2014).

II.7 Fitur Matriks Populasi Piksel

Matriks populasi piksel adalah sekumpulan sel yang terbentuk dari pembagian citra-citra membentuk kolom dan baris. Masing-masing sel berisi populasi piksel pada bagian-bagian citra yang diwakilinya. Matriks populasi piksel dapat tersusun dari matriks 2x2, 3x3, 4x4, 5x5, 6x6 dan seterusnya.

Menurut Mulyana (2006), dalam laporan tesisnya menjelaskan bahwa untuk dapat mengakomodasi penyebaran piksel obyek citra sebaiknya menggunakan matriks yang dapat mengakomodasi penyebaran piksel-piksel citra obyek tersebut.

OCR membutuhkan fitur (ciri), untuk mengenali karakter hurufnya. Ssalah satu fitur yang dapat digunakan adalah matriks populasi pixel. (Mulyana, 2014)

II.8 *L1-Metric Distance*

L1-metric melakukan pengukuran jarak antara fitur-fitur yang dimiliki dua buah citra. Dimana jarak kedua buah citra ini yang nantinya akan dipertimbangkan sebagai kemiripan antara dua buah citra. Semakin kecil nilai jarak yang dihasilkan maka kedua citra akan dianggap semakin mirip. semakin besar nilai jarak yang dihasilkan maka kedua citra akan dianggap semakin berbeda. (Theresia & Simon, 2016).

$$d(I,H)\sum_{l=1}^{n}|i_l - h_l| \quad \text{.....[1]}$$

Keterangan pada rumus 1:

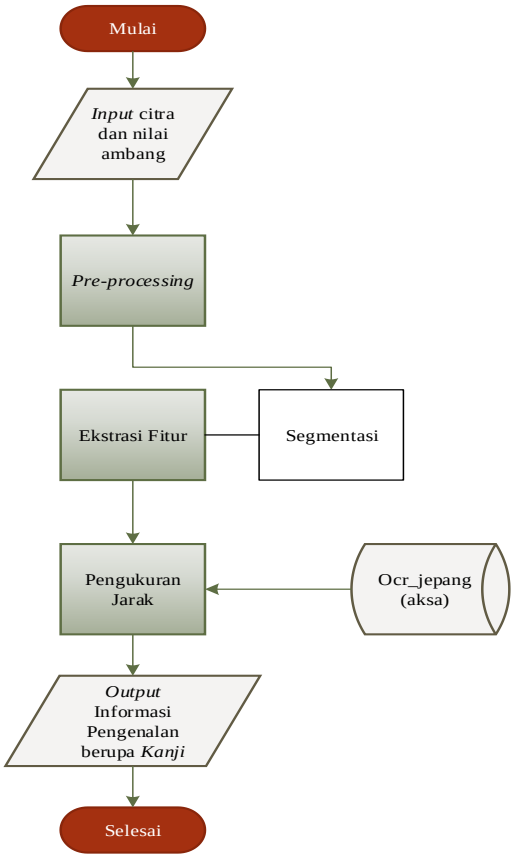
l : pencacah fitur
n : jumlah fitur
I : himpunan fitur citra pada top stack / citra yang terakhir disimpan
i : fitur citra pada top stack / citra yang terakhir disimpan
H : himpunan fitur citra yang akan diuji
h : fitur citra yang akan diuji
D(I,H) : jarak citra I terhadap citra H

III. ANALISIS DAN PERANCANGAN

III.1 Kebutuhan Data

Analisis data yang diperlukan dalam penelitian ini adalah gambar atau citra harus berekstensi .jpg, .png, .bmp dan gambar harus berupa aksara Jepang bukan aksara lain.

III.2 *Flowchart* Sistem



Gambar 4. *Flowchart* Sistem OCR

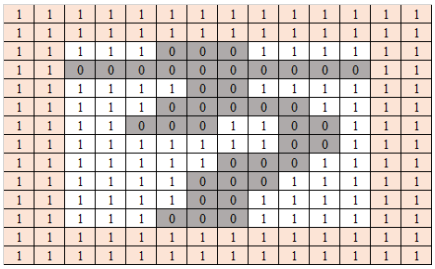
Berdasarkan Gambar 4, proses diawali dengan memasukkan citra dan nilai ambang lalu dilakukan proses *pre-processing* untuk mendapatkan citra biner dan dilakukan proses segmentasi untuk memisahkan tiap karakter dan dilanjutkan ke ekstrasi fitur untuk mendapatkan fitur yang objek yang diuji. Setelah mendapatkan fitur, dilakukan pengukuran jarak dengan *l1-metric* yang diuji dengan data sampel yang terdapat pada basis data ocr_jepang. Setelah dilakukan pengukuran jarak maka keluarlah hasil berupa informasi dari pengenalan berupa huruf *kanji*.

III.3 Ekstrasi Fitur

Ekstrasi fitur dilakukan dengan menggunakan fitur matriks populasi piksel dimana metode ekstrasi fitur merupakan cara untuk mendapatkan fitur atau ciri dari sebuah gambar atau objek citra. Pemilihan fitur matriks populasi piksel ini dikarenakan peneliti melihat penelitian sebelumnya yang telah berhasil menggunakan fitur matriks populasi piksel tersebut dan peneliti ingin mengimplementasikan fitur matriks populasi piksel tersebut untuk mengenali aksara *Hiragana* dan *Katakana*.

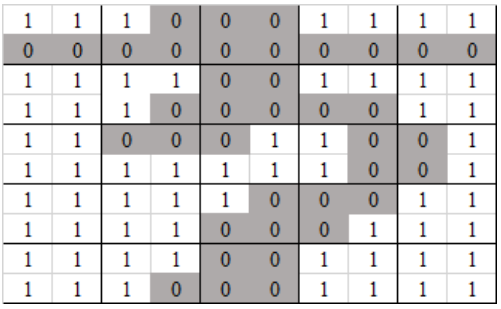
Pada ekstrasi fitur dilakukan pencarian dengan menggunakan *scanline* secara horizontal dan vertikal untuk menentukan batas atas, bawah, kiri, dan kanan dimana batas tersebut merupakan titik piksel hitam yang ditemukan pada sisi paling kanan, kiri, atas, dan bawah.

Setelah dilakukan pembatasan, maka selanjutnya citra tersebut dibagi menjadi matriks 5 x 5 dimana tiap bagian dihitung jumlah piksel hitam sehingga memperoleh 25 nilai dan juga ditambahkan perbandingan lebar dan tinggi sehingga menjadi 26 nilai. Dari 26 nilai tersebut maka dilakukan pengukuran jarak untuk pengenalan aksara yang sesuai dengan aksara yang diuji.



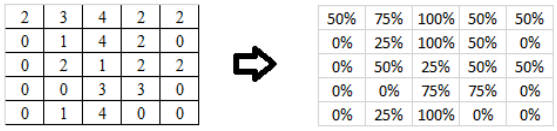
Gambar 5. Contoh Pembatasan Ekstrasi Fitur

Berdasarkan Gambar 5, ekstrasi fitur akan melakukan *scanline* dimana menemukan batas bawah, kiri, kanan dan atas. Batas tersebut ditemukan ketika menemukan nilai 0 atau piksel warna hitam yang terdapat paling kiri, kanan, atas dan bawah.



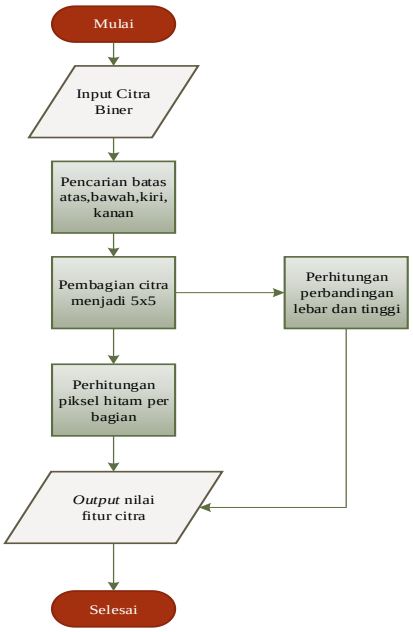
Gambar 6. Pembagian 5x5 Matriks Populasi Piksel

Setelah menentukan batas maka dilakukannya pembagian 5x5 seperti Gambar 6.



Gambar 7. Proses Matriks Populasi Piksel

Berdasarkan Gambar 7, setelah dibagi menjadi 5x5 bagian, lalu dihitung berapa banyak piksel yang terdapat tiap bagian. Dari setiap bagian tersebut maka diperoleh nilai fitur untuk citra tersebut berupa 25 nilai serta ditambah dengan perbandingan lebar dan tinggi menjadi 26 nilai fitur.

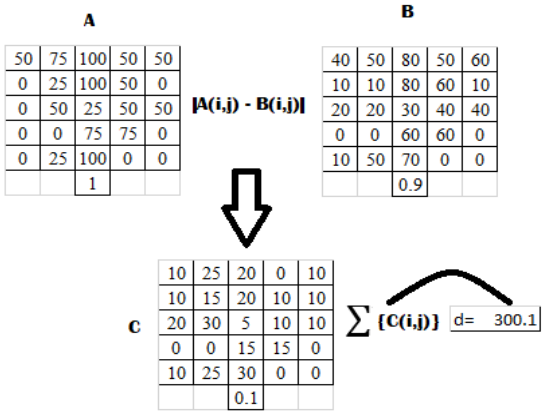


Gambar 8. *Flowchart* Proses Ekstrasi Fitur

Pada Gambar 8, proses diawali dengan masukkan citra biner dari karakter yang telah disegmentasi. Citra biner tersebut diproses untuk mencari batas kiri, kanan, atas dan bawah. Setelah itu dilakukan pembagian citra menjadi 5x5. Setelah itu dilakukan proses perhitungan piksel hitam per-bagian dan menghitung perbandingan lebar dan tinggi untuk mendapatkan fitur dari citra tersebut. Untuk proses ekstrasi fitur ini merupakan proses fitur matriks populasi piksel. (Sutoyo, 2009)

III.4 Pengukuran Jarak

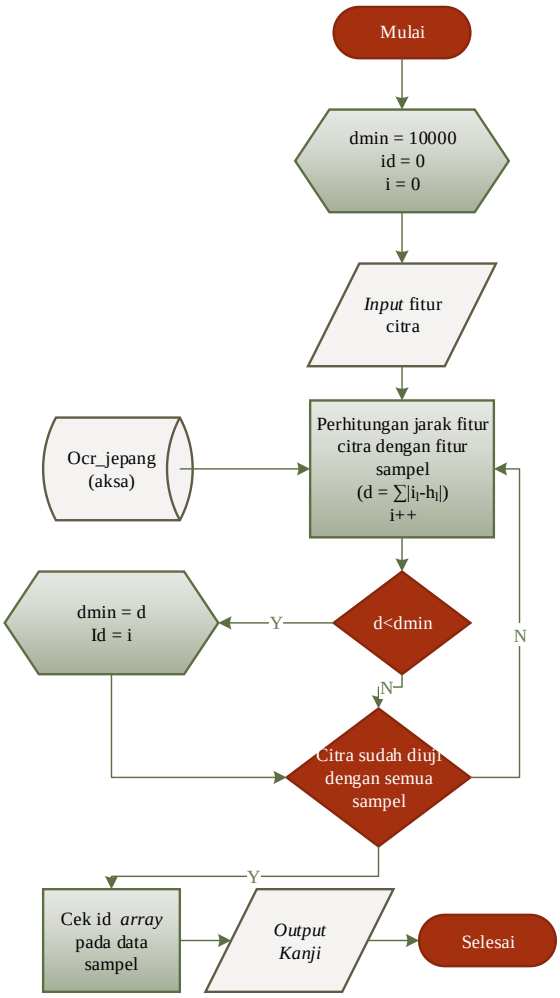
Pengukuran jarak digunakan untuk melakukan pengenalan yang mendekati nilai dari gambar yang diuji. Nilai yang didapatkan dari ekstrasi fitur digunakan dalam pengukuran jarak. Pengukuran jarak yang digunakan adalah *l1-Metric*. Peneliti menggunakan *l1-Metric* dikarenakan untuk menghitung nilai fitur yang diuji dengan fitur sampel sehingga mendapatkan nilai minimum dimana nilai yang paling kecil merupakan nilai yang terdekat dalam pengenalan aksara Jepang.



Gambar 9. Perhitungan *L1-Metric*

Berdasarkan Gambar 9, perhitungan *l1-metric* dilakukan dengan membandingkan selisih antar bagian dari 2 buah fitur yaitu fitur dari data uji dan fitur dari data sampel. Setelah diselisih maka dilakukan penjumlahan antar setiap selisihnya dan mendapatkan nilai pengukuran jaraknya. Nilai pengukuran jaraknya juga ditambahkan selisih dari 2 fitur yang dihasilkan dari perbandingan lebar dan tinggi dari citra uji dan citra sampel.

Setelah didapatkan nilai pengukuran jarak maka dilakukan pengujian dengan citra sampel yang lain dan dibandingkan nilai jarak yang terkecil. Dari nilai jarak terkecil tersebut maka ditentukan bahwa informasi data sampel tersebut sama dengan data uji.



Gambar 10. *Flowchart* Proses Pengukuran Jarak

Berdasarkan Gambar 10, proses diawali dengan inisialisasi awal dmin, id dan i dan memasukan fitur citra yang didapatkan pada proses ekstrasi fitur. Setelah itu diproses dengan pengukuran jarak *l1-metric* antara citra yang diuji dengan citra yang terdapat pada basis data. Jika yang dihasilkan dari pengukuran jarak lebih kecil dari dmin maka dmin diganti d dan id = i. Jika tidak maka dilanjutkan ke proses selanjutnya yaitu mengecek apakah citra yang diuji telah diuji dengan semua sampel atau tidak. Jika sudah diuji semua maka dilakukan pengecekan id pada *array* data sampel dan mengeluarkan hasil huruf *kanji*.

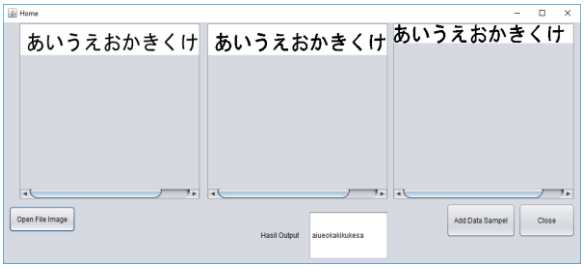
III.5 Perancangan Basis Data

Basis data atau tabel Aksa akan dijelaskan dengan kamus data berdasarkan tabel 1.

Tabel 1. Kamus Data Tabel Aksa

Nama Field	Tipe Data	Panjang	Keterangan
id	int	11	*Primary key, auto increment*
font	varchar	45	Tipe *font* aksara jepang
kanji	varchar	45	Huruf *kanji* dari aksara
aksara	varchar	45	Jenis aksara (*Hiragana, Katakana*)
fitur	varchar	5000	Fitur dari matriks populasi piksel

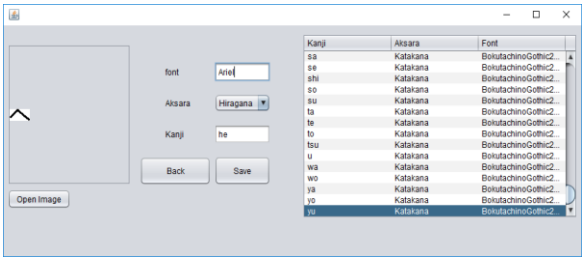
III.6 Implementasi Antarmuka



Gambar 31. Antarmuka *Recognition*

Berdasarkan Gambar 11, antarmuka pengenalan aksara Jepang dapat dimulai dengan klik *button* "*Open File Image*" untuk mengambil gambar dan langsung diproses sehingga menampilkan hasil pada *textbox* "Hasil *Output*". Untuk menambah data sampel maka dapat klik *button* "*Add* Data Sampel".

Untuk keluar program dengan klik *button* "*Close*". Ada 3 gambar pada antarmuka diatas yaitu gambar sebelah kiri adalah gambar asli, gambar tengah adalah gambar yang telah dibinerisasi, dan gambar sebelah kanan adalah gambar yang disegmentasi.



Gambar 42. Antarmuka Penambahan Sampel

Berdasarkan Gambar 12, Proses yang pertama kali dilakukan untuk penambahan sampel adalah klik *button* "*Open Image*" lalu *user* memilih gambar. Setelah memilih gambar maka langsung diproses binerisasi dan ekstrasi fitur.

Selanjutnya isi *textbox* "*font*" dengan *font* yang digunakan, jenis aksara, dan huruf *kanji*. Jika telah semua terisi maka klik *button* "*Save*" untuk menyimpan di basis data. Setelah disimpan maka akan ditambahkan di tabel sebelah kanan.

Untuk kembali ke antarmuka pengenalan aksara jepang maka dapat klik *button* "*Back*".

III.7 Implementasi Proses Binerisasi

```
private void binerisasi()
    {
        int warna,red,green,blue,abuabu;
        lebar = image.getWidth(this);
        tinggi = image.getHeight(this);
        int [] pixels = new int[lebar *
        tinggi];
        PixelGrabber pg = new
        PixelGrabber(image,0,0,lebar,tinggi
        , pixels,0,lebar);
    try
    {
            pg.grabPixels();
        }
    catch(InterruptedException ie)
    {
            System.out.println("Terjadi
            kesalahan saat mengambil data
            pixels");
            ie.printStackTrace();
            return;
    }
```

Gambar 53. Implementasi Proses Binerisasi(1)

```
for(int y=0; y<tinggi; y++){
    for(int x=0; x<lebar; x++){
        warna=pixels[y*lebar+x];
        red=(warna >> 16) & 0xff;
        green=(warna >> 8) & 0xff;
        blue=(warna) & 0xff;
        abuabu = (red + green + blue) /3;
        if(abuabu < 200){
            biner[x][y] = 0;
        }
        else{
            biner[x][y] = 255;
        }
        Color newColor = new
        Color(biner[x][y],biner[x][y],biner[x]
        [y]);
        gambarbin.setRGB(x, y,
        newColor.getRGB());
    }
}
gambarbiner.setIcon(new ImageIcon(gambarbin));
```

Gambar 64. Implementasi Proses Binerisasi(2)

Berdasarkan Gambar 13, proses yang dilakukan pertama kali sebelum mengubah citra warna ke citra biner adalah proses pengambilan piksel warna dengan penggunaan *PixelGrabber* pada *library Java*.

Berdasarkan Gambar 14, proses binerisasi dilakukan dengan mengambil warna *red*, *green*, *blue*. Lalu ketiganya dijumlahkan dan dibagi tiga sehingga mendapatkan warna abu-abu (*grayscale*).

Setelah mendapatkan warna abu-abu maka proses selanjutnya adalah membandingkan nilai abu-abu (*grayscale*) dengan nilai ambang. Nilai ambang yang dipakai adalah 200. Jika nilai abu-abu kurang dari nilai ambang maka nilai biner menjadi 0 (hitam) jika tidak maka nilai biner menjadi 255 (putih).

Proses dilakukan terus menerus sampai setiap piksel telah dibinerisasi.

III.8 Implementasi Proses Segmentasi

```
int hsegm1[] = new int[20];
int hsegm2[] = new int[20];
int jml=0; int n=0; int l=0; int min=0;
segmentasi.setIcon(new ImageIcon(gambarsegmen));
for(int x=0;x<lhasil;x++){
    int k=0;
    for(int y=0;y<thasil;y++){
        if(n==0){
            if(segm[x][y]==0){
                n=1; l=0;
                hsegm1[jml]= x;
            }
        }
        if(segm[x][y]==255)
            k= k+1;
    }
    if(l==0){
        if(k==thasil){
            n=0; l=1;
            hsegm2[jml]= x;
            jml=jml+1;
        }
    }
}
hsegm2[jml]= lhasil;
String text="";
```

Gambar 75. Implementasi Segmentasi(1)

Berdasarkan Gambar 15, proses segmentasi yang pertama dilakukan adalah menemukan titik x awal dan titik x akhir pada objek. Dengan titik x awal ditemukannya posisi awal piksel hitam yang pertama kali ditemukan dan x akhir yang ditemukan posisi akhir piksel hitam tidak ditemukan lagi.

```
for(int i=0;i<=jml;i++){
    if(hsegm2[i]-hsegm1[i]<20){
        if(i+1<=jml){
            if(hsegm2[i+1]-hsegm1[i]<33){
                hsegm2[i]=hsegm2[i+1];
                text=text + recognition
                (hsegm1[i],hsegm2[i],thasil);
                i= i+1;
            }else
                text=text
                +recognition(hsegm1[i],hsegm2[i],
                thasil);

        }else
            text=text
            +recognition(hsegm1[i],hsegm2[i],thas
            il);
    }else
        text=text +
        recognition(hsegm1[i],hsegm2[i],thasil);
}
    hasil.setText(text);
```

Gambar 86. Implementasi Segmentasi(3)

Berdasarkan gambar 16, proses ini merupakan kelanjutan dari proses pada gambar 14. Pada proses ini melakukan pembatasan antar objek dan melakukan pengukuran jarak dengan memanggil *method recognition* yang menghasilkan huruf *kanji*. Setiap *kanji* yang dihasilkan digabungkan menjadi 1 pada sebuah *text* dan kemudian ditampilkan.

III.9 Implementasi Proses Ekstrasi Fitur

```
int xmin= lebar;
int xmax = 0;
int ymin = tinggi;
int ymax= 0;
for (int y=0 ;y<tinggi; y++){
    for (int x=0; x<lebar; x++){
        if(biner[x][y]==0){
            if(xmin >=x ){xmin=x;}
            else if(ymin>=y){ymin=y;}
            else if(xmax <= x ){xmax=x;}
            else if(ymax <=y ){ymax=y;}
        }
    }
}
thasil = (ymax-ymin)+1;
lhasil = (xmax-xmin)+1;
```

Gambar 97. Implementasi Proses Ekstrasi Fitur

Berdasarkan gambar 17, proses yang pertama kali dilakukan pada ekstrasi fitur adalah dengan mencari batas kanan, kiri, atas dan bawah. Proses ini dilakukan pada antarmuka data sampel. Namun proses ini sudah masuk pada proses segmentasi pada antarmuka pengenalan aksara Jepang.

```
double matriks[][] = new double[5][5];
int lh = Math.abs(x2-x1+1);
int hsl[][] = new int[lh+2][th+2];
for(int y=0; y<th;y++){
    for(int x=0;x<lh;x++){
        hsl[x][y] = segm[x+x1][y];
    }
}
```

Gambar 108. Implementasi Matriks Populasi Piksel(1)

Berdasarkan Gambar 18, proses yang pertama kali dilakukan adalah pembuatan *variable* untuk menampung nilai dan pembagian matriks menjadi 5x5.

```
double tgmat = th/(double)5;
double lbmat = lh/(double)5;
for(int j=0;j<5;j++){
    for(int i=0;i<5;i++){
        int n=0;
        matriks[i][j]=0;
        for(int y=(int)(Math.round(j*tgmat));
        y<(int)(Math.round((j+1)*tgmat));y++){
            for(int x= (int)
            (Math.round(i*lbmat)) ;
            x<(int)(Math.round((i+1)*lbmat));
            x++){
                if(hsl[x][y] == 0){
                    matriks[i][j] +=1;
                }
                n++;
            }
        }
        matriks[i][j]=Double.parseDouble(df.form
        at((matriks[i][j]/(double)n)*100));
    }
}
double fwh = lhasil/(double)thasil;
fwh = Double.parseDouble( df.format(fwh));
```

Gambar 119. Implementasi Matriks Populasi Piksel(2)

Berdasarkan Gambar 19, proses yang menghitung jumlah piksel hitam pada setiap bagian. Lalu jumlah tersebut dibagi jumlah seluruh piksel pada bagian tersebut dan dikali dengan 100. Proses ini dilakukan terus-menerus sampai mengisi 25 bagian. Proses terakhir adalah menghitung perbandingan lebar dan tinggi.

III.10 Implementasi Proses *L1-Metric*

```
private double l1metric(double metric1[][],
double fitur1, double metric2[][], double
fitur2)
{
    double d = Math.abs(fitur2-fitur1);
    for(int i=0; i<5;i++)
    {
        for(int j=0; j<5;j++)
        {
            d += Math.abs(metric2[i][j] -
            metric1[i][j]);
        }
    }
    return d;
}
```

Gambar 20. Implementasi *L1-Metric Distance*

Berdasarkan Gambar 20, proses yang dilakukan adalah selisih antar fitur yang dijumlahkan. Selisih tersebut didapatkan berdasarkan rumus 1. Selisih yang dicari adalah selisih antara citra uji dengan citra sampel yang terdapat pada *database*.

Dari hasil pengukuran jarak tersebut maka selanjutnya akan mencari jarak yang terkecil. Dari jarak terkecilah maka citra uji itu merupakan citra sampel tersebut.

III.11 Hasil Pengujian

Jenis *font* yang menjadi data sampel didalam basis data adalah *font BokutachinoGothic2Regular, irohamaru mikami Regular*, *ＭＳ 明朝* (*Body Asian*), *Senobi Gothic Regular*, dan *Source Han Serif Medium.*

Untuk pengujiannya dilakukan dengan jenis *font* yang tidak terdapat dalam data sampel. *Font* yang digunakan untuk pengujian adalah font *Microsoft YaHei*, *SimSun*, dan *Yu Gothic Medium*. Pengujian dilakukan dengan memasukan citra aksara Jepang berupa kata dan karakter lebih dari 1.

Tabel 2. Pengujian OCR Aksara Jepang

Jumlah Aksara	Benar	Salah	Presentase
276	228	48	82,61%

Berdasarkan Tabel 2, hasil pengujian yang dilakukan menunjukan bahwa 82,61% aksara jepang dengan jenis *font* lain berhasil dikenali. Namun masih 17,39% gagal dalam mengenali aksara tersebut. Kegagalan ini dikarenakan proses binerisasi dan segmentasi yang kurang baik dimana proses binerisasi kadang menghilangkan piksel-piksel yang seharusnya tidak hilang dan segmentasi kurang mampu memisahkan per karakter.

Kegagalan ini juga didapatkan karena adanya karakter-karakter yang sama persis dengan perbedaan sedikit yang menyebabkan karakter-karakter tersebut tidak mampu dikenali.

IV. KESIMPULAN DAN SARAN

Setelah dilakukan pengujian maka penelitian ini dapat disimpulkan.

a. Matriks populasi piksel dengan menggunakan L1-Metric mampu mengenali pengenalan aksara Jepang.

b. Fitur matriks populasi piksel dapat diimplementasikan untuk mendapatkan fitur dalam pengenalan aksara Jepang.

c. *L1-metric distance* dapat diimplementasikan untuk melakukan pengukuran jarak dalam pengenalan aksara Jepang

d. Aplikasi ini dapat mengenali aksara Jepang dengan tingkat akurasi cukup tinggi.

e. Matriks populasi piksel dengan menggunakan L1-Metric distance dapat diterapkan untuk berbagai bidang dalam mengenalai karakter.

Berdasarkan penelitian ini, saran untuk peneliti selanjutnya sebagai berikut.

a. Pengujian dilakukan dengan kalimat berupa aksara Jepang

b. Dibutuhkan algoritma yang dapat melakukan segmentasi antar karakter dan berbagai ukuran *font*.

c. Menggunakan algoritma pembelajaran untuk penelitian selanjutnya.

REFERENSI

Haryanto, R. T., Mahardityawarman, R., Kusnaedi, & Priyanggodo, D. Y. (2016). Pengenalan Tulisan Tangan Karakter Jepang Menggunakan Library Tesseract Pada Android. In S. Widyarto (Ed.), *Proceeding of the 2nd Informatics Conference 2016 (ICF-2016)* (pp. 51-53). Jakarta: Universitas Budi Luhur. Retrieved from http://ojs.computing-icf.org/index.php/pic/article/view/10/21

Afriansyah, A. (2015, Desember). Analisa dan Perancangan Aplikasi Perpustakaan pada Politeknik Sekayu Menggunakan Pemrograman Java. *Jurnal Teknik Informatika Politeknik Sekayu, III*(2), 53-61. Retrieved from http://jurnal.polsky.ac.id/index.php/tips/article/view/65/19

Huyck, Christian R., Mitchell, Ian G., "Compensatory Hebbian learning for categorisation in simulated biological neural nets", Journal: Biologically InspiRed Cognitive Architectures, Vol 6, 2013, (ISSN: 2212-683X) page 3-7

Santoso, & Nurmalina, R. (2017, April). Perencanaan dan Pengembangan Aplikasi Absensi Mahasiswa Menggunakan Smart Card Guna Pengembangan Kampus Cerdas (Studi Kasus Politeknik Negeri Tanah Laut). Jurnal Integrasi, 9(1), 84-91. Retrieved from http://jurnal.polibatam.ac.id/index.php/JI/article/view/288/277

A., R. S., Erik, & Hakim, M. L. (2014). Penerapan Teknik OCR (Optical Character Recognition) pada Aplikasi Terjemahan Kitab Fiqih Safinah An-Naja Menggunakan ReadDRIS. Seminar Nasional Infromatika (pp. 60-69). Yogyakarta: UPN "Veteran". Retrieved from http://jurnal.upnyk.ac.id/index.php/semnasif/article/viewFile/994/865

Mulyana, T. M. (2014, Juni). Fitur Matriks Populasi Piksel Untuk Membedakan Frame-frame Dalam Deteksi Gerakan. JURNAL TEKNOLOGI INFORMASI, 10(1), 13-18. Retrieved from http://journal.ubm.ac.id/index.php/teknologi-informasi/article/viewFile/322/309

Sathasivam, Saratha., "Learning Rules Comparison in Neuro-Symbolic Integration", International Journal of Applied Physics and Mathematics, Vol. 1, No. 2, September 2011, AbuDhabi University, UAE (ISSN: 2010-362X) page 129-132

Septiarini, A. (2012, Juli). Segmentasi Karakter Menggunakan Profil Proyeksi. Jurnal Informatika Mulawarman, 7(2), 66-69. Retrieved from http://e-journals.unmul.ac.id/index.php/JIM/article/view/88/pdf

Sutoyo, T., mulyanto, E., Suhatono V., Nurhayanti OD., Wijanarto., Teori Pengolahan Digital, Andi Offset, Yogyakarta, 2009.

Khardon, Roni., Wachman, Gabriel., "Noise Tolerant Variants of the Perceptron Algorithm", Journal of Machine Learning Research 8 (2007) 227-248 Submitted 11/05; Revised 10/06; Published 2/07

Sathasivam, Saratha., "Learning Rules Comparison in Neuro-Symbolic Integration", International Journal of Applied Physics and Mathematics, Vol. 1, No. 2, September 2011, AbuDhabi University, UAE (ISSN: 2010-362X) page 129-132