GABOR WAVELET DAN SUPPORT VECTOR MACHINE
TEORI DAN PRAKTEK

Mangaras Yanu F., S.T., M.Eng.
Indah Reforsiana Nurjanah, S.Kom
Hari Prapcoyo, M.ICT
Rifki Indra Perwira, S.Kom., M.Eng.

Lembaga Penelitian & Pengabdian Masyarakat
Universitas Pembangunan Nasional "Veteran" Yogyakarta

GABOR WAVELET DAN SUPPORT VECTOR MACHINE

TEORI DAN PRAKTEK

Penulis :

Mangaras Yanu Florestiyanto
Indah Reforsiana Nurjanah
Hari Prapcoyo
Rifki Indra Perwira

GABOR WAVELET DAN SUPPORT VECTOR MACHINE

TEORI DAN PRAKTEK

Mangaras Yanu Florestiyanto, S.T., M.Eng.
Indah Reforsiana Nurjanah, S.Kom
Hari Prapcoyo, M.ICT
Rifki Indra Perwira, S.Kom., M.Eng.

ISBN **: 9 786236 896747**

Diterbitkan oleh :
LPPM
UPN Veteran Yogyakarta
Jln. SWK 104 (Lingkar Utara) Condong Catur, Yogyakarta 55283
Telp. 0274 486188, 486733, Fax : 0274 486400

KATA PENGANTAR

Puji syukur kehadapan Tuhan Yang Maha Kuasa bahwa atas petunjukNya maka buku ajar ini bisa disusun. Pada awalnya buku ini merupakan hasil riset, namun ternyata setelah mengalami berbagai macam penyesuaian buku ini juga dapat digunakan untuk buku ajar. Untuk itu materi bahan ajar yang mudah dipahami, serta memudahkan untuk melakukan rekonstruksi pembelajaran.

Gabor Wavelet dan Support Vector Machine merupakan bahan ajar yang ditulis bagi mereka yang ingin mendapatkan pemahaman dan pengetahuan mengenai konsep dasar serta implementasinya pada indentifikasi huruf braille. Tidak dipungkiri bahwa pendampingan belajar dirumah merupakan hal yang sangat penting dalam memahami perkembangan belajar, tak terkecuali untuk penyandang tunanetra. Namun, tidak semua orang memiliki kemampuan untuk membaca huruf braille, sehingga akan menimbulkan permasalahan dalam pendampingan belajar.

Kami yakin masih banyak kekurangan yang terjadi dalam bahan ajar ini. Oleh karena itu kritik atau saran penyempurnaan kami harapkan dari para pembaca buku ini. Semoga usaha yang mulia untuk menjadikan pendidikan di negara kita ini bermutu tinggi dan melahirkan Sumber Daya Manusia yang bermutu selalu mendapatkan petunjukNya. Terimakasih

Yogyakarta, Oktober 2021

Tim penulis

DAFTAR ISI

PENDAHULUAN

Pertukaran informasi pada era digital terutama dalam bentuk visual dan cetak sudah umum dilakukan oleh orang dengan penglihatan normal namun berbeda dengan seseorang yang memiliki kekurangan dalam penglihatan atau tunanetra. Perbedaan tersebut akan menimbulkan permasalahan dalam bertukar informasi dalam bentuk tulisan. Sebagai contohnya dalam pendampingan dari seorang tunanetra oleh keluarga dengan berpenglihatan normal.

Pada aspek pendidikan, belajar tidak hanya dilakukan di sekolah, namun juga di rumah. Dalam hal ini, orang tua sangat memiliki peran penting dalam membantu anak memahami perkembangan belajar mereka. Seorang tunanetra dalam menulis dan membaca menggunakan sistem huruf braille, tentu tidak semua orang dengan penglihatan normal memiliki kemampuan untuk membaca huruf braille tersebut. Hal ini menyebabkan para orang tua mengalami kesulitan dalam mendampingi seorang anak tunanetra dalam belajar.

Sejauh ini tulisan braille diterjemahkan dengan meraba huruf- huruf braille untuk dapat memahami maksut tulisan tersebut. Sedangkan, untuk orang dengan penglihatan normal yaitu dengan cara mencocokkan huruf braille satu per satu dengan huruf

abjad hingga dapat terbaca.

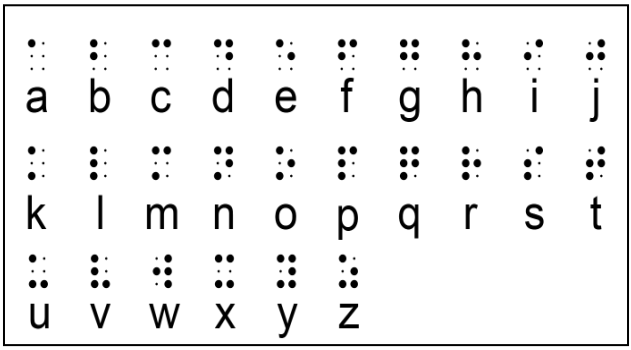
Huruf braille merupakan tulisan sentuh yang terdiri dari serangkaian titik timbul yang menunjukkan huruf, angka, dan simbol. Huruf braille tersusun dari enam titik yaitu dua titik secara horizontal dan tiga titik secara vertikal. (Harimi, 2018) Ukuran standar sebuah karakter Braille adalah sekitar 4 mm lebar dan 6 mm tinggi dengan ketebalan sekitar 0,4 mm. Dalam penulisannya, memperkecil ataupun memperbesar ukuran huruf tersebut dapat mempengaruhi tingkat keterbacaannya oleh ujung-ujung jari para tunanetra. (Tarsidi, 2000)

Dalam proses penulisan huruf menggunakan alat tulis yang berbeda yaitu *reglet* yang berupa papan cetak berlubang yang dijepit pada kertas tulis kemudian menggunakan paku tumpul sebagai pencetak huruf braille. Agar titik-titik timbul tersebut dapat bertahan lama, maka ketebalan kertas menjadi hal yang harus diperhatikan.

1.1. Huruf Braille

Huruf braille merupakan tulisan sentuh yang terdiri dari serangkaian titik timbul yang menunjukkan huruf, angka, dan simbol. Huruf braille tersusun dari enam titik yaitu dua titik secara horizontal dan tiga titik secara vertikal. (Harimi, 2018). Huruf braille digunakan oleh para penyandang tuna netra untuk mengembangkan wawasannya. Sistem penulisan ini memiliki satuan dasar yaitu sel braille. Satu sel braille terdiri dari 6 titik

yang dapat dalam kondisi aktif (timbul) atau tidak aktif (datar). Dengan kombinasi keenam titik ini dapat membentuk 63 macam kombinasi/karakter yang berupa huruf, angka, tanda baca, musik, operator-operator dasar matematika dan beberapa tanda singkatan. (Tarsidi, 2000).



Gambar 1. Huruf braille alphabet

1.2. Ekstraksi Ciri

Ekstraksi ciri atau fitur merupakan proses pengambilan informasi penting yang ada pada suatu citra. Ciri disini ialah karakteristik yang mencirikan suatu objek. Ekstraksi ciri bertujuan mengambil karakteristik khusus dari suatu objek untuk kemudian disimpan. Ciri - ciri inilah yang nantinya akan digunakan untuk pembanding dalam mengenali objek tertentu

dalam suatu citra. Pada penelitian ini metode yang digunakan untuk mengekstraksi ciri yaitu metode *Gabor Wavelet.*

1.3. Gabor Wavelet

Gabor Wavelet merupakan salah satu dari jenis *wavelet* yang ada. *Wavelet* berasal dari kata *wave* dan *let* yang berarti gelombang yang pendek, yaitu gelombang yang mempunyai durasi sangat pendek atau terbatas. Dengan kata lain fungsi gelombang yang akan dijadikan objek dilokalisasi. Teknik ekstraksi ciri menggunakan fungsi Gabor *Wavelet* digunakan untuk mengekstrak ciri dari citra yang ternormalisasi. Fungsi 2-D *Gabor Wavelet* merupakan tapis spasial pelewat bidang yang optimum dalam meminimalisasi ciri yang tidak penting dalam kawasan spasial dan frekuensi. (Rangayyan, 2007).

Kelebihan metode Gabor Wavelet yaitu memiliki kemampuan yang dengan dapat mudah menyesuaikan lokalisasi detail pada domain spasial dan frekuensi serta kemiripannya dengan representasi frekuensi dan orientasi pada sistem visual manusia sehingga metode ini sangat popular dan dapat menghasilkan hasil yang baik pada area segmentasi tekstur, pengenalan sidik jari, dan pengenalan wajah (Yong-zhao, Jing- fu, De-jiao, & Peng, 2004). Persamaan *Gabor Wavelet* dapat

dilihat sebagai berikut (Adak, 2014):

$$g(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} exp\left(-\frac{1}{2}\left[\frac{x'^2}{\sigma_x^2} + \frac{y'^2}{\sigma_y^2}\right]\right) cos(2\pi F x') \text{ .....(1.1)}$$

$$g(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} exp\left(-\frac{1}{2}\left[\frac{x'^2}{\sigma_x^2} + \frac{y'^2}{\sigma_y^2}\right]\right) sin(2\pi F y') \text{ .....(1.2)}$$

$$x' = x\cos\theta + y\sin\theta \text{ .....(1.3)}$$

$$y' = -x\sin\theta + y\cos\theta \text{ .....(1.4)}$$

$$output = \sqrt{r^2 + im^2} \text{ .....(1.5)}$$

Keterangan:

F = frekuensi = $\frac{\sqrt{2}}{2^0}, \frac{\sqrt{2}}{2^1}, \frac{\sqrt{2}}{2^2}, \ldots$

θ = orientasi atau sudut pandang terhadap *image* berkisar dari 0 hingga 2π

σ = sigma

Pada persamaan-persamaan diatas, dibagi menjadi dua yaitu kernel riil dan kernel imajiner. Persamaan 1.1 merupakan kernel riil dan persamaan 1.2 merupakan kernel imajiner. Sedangkan untuk persamaan 1.5 merupakan rumus yang digunakan untuk menghitung hasil dari konvolusi antara citra dengan kernel riil dan citra dengan kernel imajiner.

Proses *filtering image* menggunakan *Gabor Wavelet* merupakan operasi konvolusi antara matriks citra dengan matriks kernel. Parameter-parameter seperti *x*, *y*, *F*, dan θ digunakan untuk membangun kernel konvolusi real dan imajiner. Kernel konvolusi inilah yang merupakan filter *Gabor* disisi real dan imajiner. Operasi konvolusi

antara citra dengan kernel konvolusi imajiner akan menghasilkan citra imajiner (*im*), sedangkan operasi konvolusi antara citra dengan kernel konvolusi real akan menghasilkan matriks citra real (*re*). Kedua matriks tersebut memiliki dimensi yang sama seperti matriks citra. Citra imajiner dan citra real kemudian dilakukan proses magnitude untuk mendapatkan matriks yang berisi nilai absolut dari citra. (Wisesty & Mutiah, 2016)

Ciri yang diekstrak pada contoh penerapan buku ini merupakan ciri tekstur yang diantaranya adalah standar deviasi, ciri mean, ciri median, dan variansi. Ciri diekstrak dari matrik absolut. Ciri tersebut merupakan satu nilai yang didapat dari kombinasi tiap nilai sudut orientasi θ dan nilai frekuensi F. Kombinasi nilai θ dan F yang berbeda akan menghasilkan nilai ciri yang berbeda pula. Nilai-nilai ciri ini merupakan elemen- elemen pada satu vektor ciri *image*. Dimensi vektor ciri yang dihasilkan tergantung pada jumlah kombinasi θ dan F yang digunakan. (Wisesty & Mutiah, 2016)

1.4. Support Vector Machine

Support Vector Machine merupakan metode pembelajaran yang digunakan untuk klasifikasi biner, ide dasarnya adalah mencari *hyperplane* terbaik yang berfungsi sebagai pemisah dua kelas pada *input space.* (Boswell, 2002). Pada metode SVM akan dilakukan pencarian dan pemisahan *hyperplane* yang berada pada 2 kelas. *Hyperplane* yang menjadi pemisah terbaik dapat ditemukan melalui margin *hyperplane* dan mencari titik maksimalnya. Margin merupakan jarak antara *hyperplane* dengan *pattern* yang terdekat dari tiap kelas. *Pattern* yang paling dekat itulah yang disebut *Support Vector.*

SVM dapat diterapkan pada data dengan persebaran linier maupun non-linier. Data yang terpisah secara linier berarti data mengelompok dengan baik sesuai dengan label. Proses pembelajaran pada SVM bertujuan untuk mendapatkan hipotesis berupa bidang pemisah terbaik yang tidak hanya meminimalkan *empirical risk* yaitu rata-rata *error* pada data pelatihan, tetapi juga memiliki generalisasi yang baik.

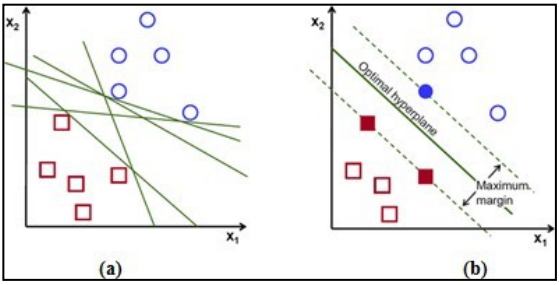
Menurut (Nugroho, 2003), karakteristik SVM secara umum dirangkum sebagai berikut:

1. Secara prinsip SVM adalah linear classifier.
2. Pattern recognition dilakukan dengan mentransformasikan data pada ruang input (input

space) ke ruang yang berdimensi lebih tinggi (feature space), dan optimisasi dilakukan pada ruang vector yang baru tersebut. Hal ini membedakan SVM dari solusi pattern recognition pada umumnya, yang melakukan optimisasi parameter pada hasil transformasi yang berdimensi lebih rendah daripada dimensi input space.

3. Menerapkan strategi Structural Risk Minimization (SRM).
4. Prinsip kerja SVM pada dasarnya hanya mampu menangani klasifikasi dua kelas, namun telah dikembangkan untuk

Klasifikasi lebih dari dua kelas dengan adanya pattern recognition.



Gambar 1.2 Model data dengan berbagai macam *hyperplane* (a), model data dengan *hyperplane* terbaik (b)

Pada gambar 1.2(a) merupakan model data dengan berbagai macam *hyperplane* dimana terdapat beberapa *hyperplane* yang memisahkan *dataset* sesuai kelasnya. Sedangkan pada gambar 1.2(b) merupakan model data dengan *hyperplane* terbaik yang memiliki margin (jarak dua set objek dari dua kelas berbeda) maksimal. *Hyperplane* terbaik merupakan *hyperplane* yang berada diantara dua set objek dari dua kelas (Cortes, Vapnik, C., & Vladimir, 1995)

Menurut Boswell (2002) diberikan contoh {xi,yi}, i= 1, 2 , … ,l. dimana setiap contoh memiliki input d ($x_i \in R^d$) dan label kelas dengan salah 1 dari 2 nilai ($y_i \in \{-1,1\}$) . Sekarang semua *hyperlines* di R^d diparameterisasi oleh vektor (w), dan konstanta (b), dinotasikan dalam persamaan :

$w.\ x + b = 0$ ……..…..(1.6)

Pattern x_i yang termasuk kelas –1 (sampel negatif) dapat dirumuskan sebagai pattern yang memenuhi pertidaksamaan :

$w.x_i + b \leq -1$…….....…….....……..(1.7)

sedangkan untuk pattern x_i yang termasuk kelas +1 (sampel positif)

$w.x_i + b \leq +1$.....(1.8)

Secara umum formula yang digunakan dalam SVM adalah sebagai berikut :

$f(x) = sign\ (w.\ x + b)$ …….....…….....(1.9)

Keterangan :

w = vector beban

b = threshold

x = input pola

Fungsi $sign$ ( ) merupakan fungsi normalisasi, jika nilai x didalam fungsi $sign\ (x)$ adalah lebih besar dari 0 maka fungsi memberikan nilai 1. Jika nilai x didalam fungsi $sign\ (x)$ adalah lebih kecil dari 0 maka fungsi memberikan nilai -1.

Untuk mengatasi permasalahan data yang tidak dapat diklasifikasikan secara linier diselesaikan menggunakan ”kernel trick”. Syarat sebuah fungsi untuk menjadi fungsi kernel adalah memenuhi teorema Mercer yang menyatakan bahwa matriks kernel yang dihasilkan harus bersifat positive semi- definite. Fungsi kernel yang

umum digunakan adalah sebagai berikut (Sembiring, 2007):

a. *Linier Kernel*

$K(x_i, x) = x_i^T x$ ……).……..……..…(1.10)

Kernel *linier* merupakan jenis kernel yang digunakan untuk menangani masalah data yang *linier*. Kernel ini sesuai untuk mengklasifikasikan data dimana data tersebut dapat dipisahkan secara *linier* oleh sebuah hyperplane Dengan x_i merupakan data latih, dan x merupakan data uji

b. *Polynomial Kernel*

$K(x_i, x) = (\gamma . x_i^T x + r)^p, \gamma > 0$…..……..(1.11)

Kernel polynomial merupakan fungsi kernel yang umum digunakan pada algoritma Support Vector Machine untuk menangani data yang tidak dapat dipisahkan secara linier. Kernel ini merepresentasikan kesamaan vektor pada feature space. Pembentukan model menggunakan kernel polynomial tidak hanya berdasarkan kesamaan vektornya saja tetapi memperhatikan juga kombinasi antar vektor yang biasa disebut interaction feature. Dengan x_i merupakan data latih, x merupakan data uji, *p* adalah derajat polynomial, γ adalah *scalling parameter* dari jarak Euclidean dan *r* adalah nilai konstanta.

c. *Radial Basis Function Kernel*

$K(x_i, x) = exp(-\gamma|x_i - x|^2), \gamma > 0$…….....(1.12)

Kernel *Radial Basis Function* atau kernel gaussian merupakan kernel yang secara *nonlinear* memetakan sampel ke dalam ruang dimensi yang lebih tinggi, sehingga tidak seperti kernel linear, kernel ini dapat menangani kasus ketika hubungan antara label kelas dan atributnya tidak linear. Dengan $x_)$ merupakan data latih, x adalah data uji, γ adalah *scalling parameter* dari jarak Euclidean.

d. Sigmoid *Kernel*

$K(x_i, x) = tanh(\gamma . x_i^T x + r)$ .....)..... (1.13)

Kernel sigmoid merupakan kernel yang digunakan untuk memecahkan masalah quadratic programming dengan yang dapat dipisahkan secara nonlinier. Kernel ini menggunakan fungsi sigmoid bipolar yang digunakan sebagai fungsi aktivasi untuk neuron buatan. Dengan $x_)$ merupakan data latih, x adalah data uji, γ adalah *scalling parameter* dari jarak Euclidean dan *r* adalah nilai konstanta

Metode sekuential training merupakan metode yang digunakan untuk training data agar menghasilkan hyperplane yang optimal. Metode ini digunakan juga untuk mendapatkan nilai α. Metode ini memiliki langkah proses seperti berikut (Jayanti, Novianti, & Sumalya, 2017) :

1. Inisiasi α_i = 0
2. Menghitung persamaan matriks hessian dengan persamaan 1.14

$$D_{ij} = y_i y_j (K(x_i x_j) + \lambda^2) \quad \text{.....(1.14)}$$

3. Lakukan tahap (a), (b) dan (c) dibawah untuk = 1, 2, 3, .. , n
   a. Menghitung nilai Ei *(error)* menggunakan persamaan 1.15.

$$E = \sum_{j=1}^{n} \alpha_j D_{ij}) \quad \text{.....(1.15)}$$

   b. Menghitung nilai $\delta\alpha_i$

$$\delta\alpha_i = \min[\max[\gamma(1-E_i), -\alpha_i], C-\alpha_i] \quad \text{..... (1.16)}$$

   c. Memperbarui nilai α_i

$$\alpha_i = \alpha_i + \delta\alpha_i \quad \text{.....(1.17)}$$

E_i : Nilai error

D_{ij} = Nilai elemen matriks hessian ke-ij

y_i = kelas data ke-i

y_j = kelas data ke-j

γ = Gamma

$\delta\alpha_i$ = Delta alpha i

λ = batas teoritis yang akan diturunkan

4. Proses 1,2 dan 3 diulangi hingga $\delta\alpha_i$ mencapai konvergen (tidak ada nilai signifikan).
5. Menghitung nilai $w.x^+$ dan $w.x^-$ untuk mendapatkan nilai bias menggunakan persamaan berikut.

$$w.\mathrm{x}^+ = \alpha_i y_i K(x, \mathrm{x}^+) \quad \text{.....(2.34)}$$

$$w.\mathrm{x}^- = \alpha_i y_i K(x, \mathrm{x}^-) \quad \text{.....(2.35)}$$

$$b = -\frac{1}{2}(w.\mathrm{x}^+ + w.\mathrm{x}^-) \quad \text{.....(2.36)}$$

Keterangan:

$K(x,\mathrm{x}^+)$: Nilai kernel data x dengan data x kelas positif yang memiliki nilai α tertinggi.

$K(x,\mathrm{x}^-)$: Nilai kernel data x dengan data x kelas negatif yang memiliki nilai α tertinggi.

b : Nilai bias

Pada awalnya SVM dikembangkan unyuk mengatasi persoalan dua kelas. Dan kemudian dikembangkan kembali untuk mengklasifikasi lebih dari dua kelas atau multikelas (Santosa, 2007). Klasifikasi multikelas dibagi menjadi dua yaitu satu lawan semua atau *one against all* (OAA) dan satu lawan satu atau *one against one* (OAO) (Sembiring, 2007).

1. Metode *One Against All* (OAA)

Dengan metode *one against all* (OAA) atau satu lawan semua, tahap klasifikasi awal dengan menentukan banyaknya kelas (k) dan k merupakan fungsi pemisah. Setiap model klasifikasi ke-i akan dilatih dengan menggunakan keseluruhan data untuk mencari solusi permasalahan dengan memberi label +1 pada kelas ke-i dan seluruh sampel diberi lainnya dengan label -1. Sebagai contoh, diberikan permasalahan klasifikasi dengan 3 kelas yaitu kelas 1, kelas 2, dan kelas 3. Jika akan diujikan, untuk kelas 1 semua data di kelas 1 diberi label +1 dan data kelas lainnya -1, untuk kelas 2 semua data di kelas 2 diberikan label +1 dan kelas lainnya diberi label -1, begitu juga untuk kelas 3, semua data di kelas 3 diberi label +1 dan kelas lainnya diberi label -1. Setelah itu, melakukan pencarian *hyperplane* dengan algoritma SVM dua kelas dan akan didapatkan *hyperplane* untuk setiap kelasnya. Kemudian menentukan kelas dari data baru (x) berdasarkan nilai

hyperplane terbesar dengan menggunakan persamaan sebagai berikut. (Hsu & Lin, 2002).

$$kelas\ x = argmax(\sum_{i=1}^{m} \alpha_i y_i K(x, x_i) + b \quad \text{.....(1.21)}$$

2. Metode *One Against One* (OAO)

Metode satu lawan satu atau *one against one* (OAO) dibangun dengan $k(k-1)$ buah model klasifikasi biner dengan k merupakan jumlah kelas. Setiap model klasifikasi akan dilatih pada data dari dua kelas (Santosa, 2007). Sebagai contoh, diberikan permasalahan klasifikasi dengan 3 kelas, yaitu kelas 1 , kelas 2, dan kelas 3. Ketika 1 dilakukan proses *training*, semua sampel pada kelas 1 diberikan label +1 dan semua sampel pada kelas 2 diberi label negative (-1) dan selanjutnya dilakukan dengan cara yang sama.

IMPLEMENTASI

2.1. Contoh Penerapan

Gabor Wavelet merupakan salah satu metode untuk mengektraksi ciri dari suatu citra. Gabor Wavelet memiliki dua kernel yaitu kernel imajiner dan kernel riil. Citra input akan dikonvolusikan dengan masing-masing kernel yang selanjutnya menghasilkan citra imajiner dan citra riil. Kedua matriks citra tersebut memiliki dimensi yang sama dengan dengan citra awal. Kemudian matriks citra imajiner dan matriks citra riil dilakukan proses perhitungan nilai magnitude untuk mendapatkan matriks yang berisi nilai absolut dari citra. Dari matriks magnitude tersebut selanjutnya akan dilakukan proses ekstraksi. Ciri yang diekstrak pada penelitian ini merupakan ciri tekstur yang diantaranya adalah standar deviasi, ciri mean, ciri median, dan variansi.

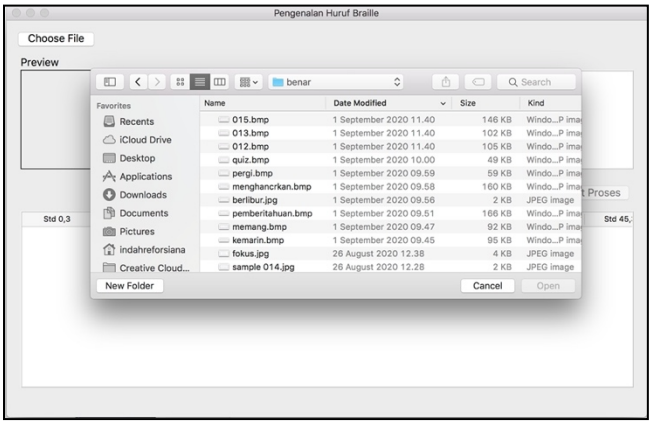
Support Vector Machine merupakan salah satu dari metode yang dapat digunakan untuk mengklasifikasikan suatu data. Dalam proses klasifikasi terdapat dua data, yaitu data latih dan data uji. Data latih bertujuan untuk melatih algoritma, sedangkan untuk data uji bertujuan untuk menguji tingkat keberhasilan dalam mengklasifikasikan data baru berdasarkan algoritma yang sebelumnya sudah dilatih.

Berikut ini adalah contoh penerapan dari ekstraksi ciri Gabor Wavelet dan klasifikasi Support Vector Machine dalam mengidentifikasi huruf braille menjadi

huruf abjad. Huruf braille yang digunakan pada aplikasi ini berupa citra huruf braille tiap kata. Kemudian dilakukan tahapan *image processing* dilakukan pada citra untuk mempersiapkan citra untuk selanjutnya disegmentasi dan mengekstraksi cirinya menggunakan *Gabor Wavelet*. Dari hasil ekstraksi ciri tersebut, akan disimpan sebagai data latih dan data uji untuk diklasifikasikan dengan menggunakan metode *Support Vector Machine.* Berikut tampilan aplikasi, penjelasan dan *Source Code* dari aplikasi tersebut:

2.2. Halaman Input Citra

Pada tahap input citra dilakukan dengan menekan tombol *choose file*. Citra uji yang dapat dimasukkan ke dalam aplikasi adalah file yang berekstensi JPG dan BMP. Ketika tombol *choose file* ditekan, maka akan menampilkan jendela baru berupa direktori file citra yang ingin diproses. Kemudian citra yang terpilih akan ditampilkan pada label *preview* gambar. Proses input terdapat pada modul program 1.1 dan tampilan antarmuka pada gambar 1.3 berikut.



Gambar 1.3. Pilih gambar

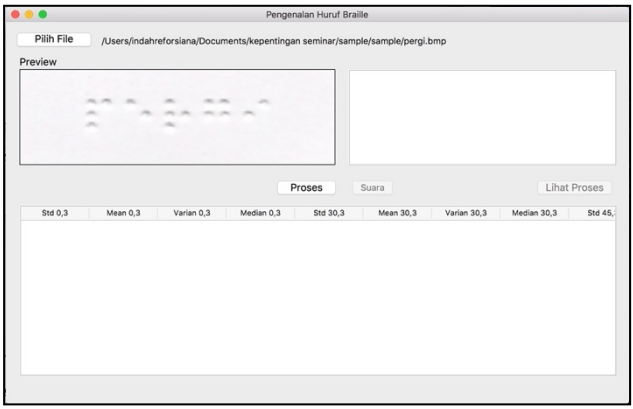
Berikut ini merupakan pseudocode input citra braille.

```
def mybutton_clicked(self):
        options = QFileDialog.Options()fileName =
QFileDialog.getOpenFileName(self,"QFileDialog.getOpenFileName()",
"","Image  Files  (*.jpg  *.bmp  *.png)",options=options)
        if fileName:
            print(fileName[0]) global pic_path
            pic_path = fileName[0]
            self.namafl.setText(fileName[0])
            pixmap=QtGui.QPixmap(fileName[0])
self.tampilfl.setPixmap(pixmap.scaled(461,141,transform
Mode=QtCore.Qt.SmoothTransformation))
        df1 = pd.read_csv("header.csv")
       self.textBrowser.setText(" ") model =
       DataFrameModel(df1)
       self.tableView.setModel(model)
       self.pushButton.setEnabled(False)
       self.btnsuara.setEnabled(False)
```

Modul 1.1. Input citra

Jika file citra sudah terpilih, maka dari variabel filename tersebut akan ditampilkan sebagai text dan juga gambar. Untuk menampilkan sebagai text, akan ditampilkan pada label namafile sehingga *user* dapat

mengetahui alamat atau *path* dari file citra. Kemudian ditampilkan pada label tampilfile sebagai gambar *preview* sehingga *user* dapat melihat gambar yang dipilih sebelum dilakukan pemrosesan. Pada saat sebelum melakukan proses, untuk memastikan *form* masih kosong, maka pada tampilan tabel akan menampilkan data csv yang berupa header dari data uji dan textBrowser diberikan isi spasi atau kosong. Dan juga tombol untuk melihat proses dan tombol suara masih bernilai *false.* Berikut ini tampilan antarmuka dengan *file* citra yang sudah dipilih terdapat tampilan dari citra dan juga alamat dari *file* citra. Untuk memproses citra tersebut, maka *user* dapat menekan tombol proses.



Gambar 1.4 Tampilan Buka Gambar

2.3. Tampilan Hasil Identifikasi

Pengolahan citra dilakukan untuk mempersiapkan citra untuk diolah pada proses segmentasi. Pada pengolahan citra terdiri dari beberapa proses, yaitu

grayscale, median filter, citra biner, dilasi citra dan erosi citra. Citra yang akan diproses berdasarkan alamat file yang disimpan pada variabel pic_path. Proses pengolahan citra akan berjalan dengan cara menekan tombol proses pada aplikasi.

Fungsi pengolahan citra terdapat pada modul program 1.2. Tahap pertama fungsi akan membaca path citra dengan menggunakan pic = cv2.imread(pic_path) citra yang dibaca akan di *rezise* ukurannya menjadi dua kali lebih besar untuk memudahkan dalam mengolah citra. Tahap kedua mengubah citra menjadi *grayscale image,* citra akan disimpan dengan nama grayIMG.jpg pada folder yang sama dengan file aplikasi. Tahap selanjutnya memperbaiki kualitas citra dengan median filter, hasil citra akan disimpan dengan nama filterMed.jpg pada folder yang sama dengan file aplikasi. Kemudian tahap citra biner dilakukan dengan menggunakan citra median filter dan citra *grayscale.* Citra *grayscale* digunakan sebagain acuan dalam menentukan ambang batas berdasarkan rata-rata dari derajat keabuan citra *grayscale*. Kemudian membuat kanvas citra yang berisi piksel 0 dengan ukuran yang sama dengan citra yang sudah di *resize.* Kanvas tersebut digunakan untuk menyimpan piksel yang telah proses. Piksel pada citra biner yang letaknya dideteksi oleh perulangan berdasarkan tinggi dan lebar citra grayscale yang selanjutnya melakukan pengecekan kondisi apabila pada titik piksel tersebut nilai citra

median filter kurang dari batas bawah maka akan diubah menjadi 0 atau hitam dan sebaliknya jika lebih dari batas bawah maka akan diubah menjadi 255 atau warna putih. Hasil citra biner akan disimpan dengan nama binerIMG.jpg pada folder yang sama dengan file aplikasi. Setelah itu dari citra biner untuk menghilangkan titik kecil yang mengganggu, piksel yang berwarna putih akan diproses menjadi citra dilasi dan akan disimpan dengan nama dilImg.jpg pada folder yang sama dengan file aplikasi. Dan untuk mengembalikkan titik braille yang penting ke ukuran semula, maka dari citra dilasi tersebut piksel yang berwarna hitam akan diproses menjadi citra erosi, citra akan disimpan dengan nama erosiImg.jpg pada folder yang sama dengan file aplikasi.

```
def proses_clicked(self):
        QtWidgets.QApplication.processEvents()pic =
        cv2.imread(pic_path)
        brs = int(pic.shape[1]*2)klm =
        int(pic.shape[0]*2)
        crop = cv2.resize(pic, (brs, klm))


        img_gray                =            cv2.cvtColor(crop,
cv2.COLOR_BGR2GRAY)
          cv2.imwrite('grayIMG.jpg', img_gray)

          imgFilterMedian = cv2.medianBlur(img_gray,5)
```

```
          cv2.imwrite('filterMed.jpg', imgFilterMedian)

          jum = 0
          count = len(img_gray) * len(img_gray[0])for i in
          range (len(img_gray)):
              for j in range (len(img_gray[0])):jum +=
                  img_gray[i,j]
                  meanImg = jum/count

          batasbawah = meanImg-20

          print("min= "+str(meanImg))
        img_biner             =          np.zeros(crop.shape,
dtype=np.uint8)
          for i in range (len(img_gray)):
              for j in range (len(img_gray[0])):

                  if imgFilterMedian[i,j]<=batasbawah :
                      img_biner[i,j] = 0
                  else:
                      img_biner[i,j] = 255

          cv2.imwrite('binerIMG.jpg', img_biner)

          kernel  =  np.ones((3,3),  np.uint8)  #tipedata
uint
          dilation                                                =
cv2.dilate(img_biner,kernel,iterations = 1)
        cv2.imwrite("dilImg.jpg", dilation)

        erosiImg    =    cv2.erode(dilation,    kernel,iterations
= 1)
        cv2.imwrite("erosiImg.jpg",erosiImg)
```

Modul Program 1.2. Pengolahan citra

Setelah itu dilakukan tahap normalisasi kontur yang dilakukan dengan mengubah titik braille dengan bentuk dan ukuran yang sama sehingga dapat meningkatkan tingkat pembacaan. Tahap ini dimulai dengan proses deteksi tepi berdasarkan citra erosi. Setelah pemrosesan citra deteksi tepi, proses dilanjutkan dengan mencari *contour* citra dan selanjutnya mencari titik tengah berdasarkan kontur yang didapatkan dengan memanfaatkan fungsi *moments invariant*. Setelah mendapatkan koordinat titik tengah kontur, membuat

lingkaran dengan titik tengah pada setiap kontur. Kemudian mencari area seluruh kontur citra berdasarkan citra yang konturnya sudah dinormalisasi. Dari kontur tersebut digabungkan seluruh konturnya kemudian mencari nilai titik x,y dan panjang serta tinggi melalui x, y, w, h = cv2.boundingRect(contour) yang merupakan kotak yang mengelilingi seluruh area kontur. Selanjutnya memotong citra berdasarkan titik x,y sebagai titik pertama dan (y+h, x+w) sebagai titik kedua. Tahap selanjutnya proses invert, yaitu membalikkan warna putih menjadi hitam dan warna hitam menjadi putih. Berikut pseudocode normalisasi citra pada modul program berikut.

```
erosi = cv2.imread("erosiImg.jpg")
edges = cv2.Canny(erosi,25,255,L2gradient=False)
cv2.imwrite("detCanny.jpg", edges)
img = edges.copy() image
= edges.copy()
contours,    hierarchy    =    cv2.findContours    (img,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
for cnts in contours:
   M = cv2.moments(cnts)if
   (M["m00"]!=0):
     cX = int(M["m10"] / M["m00"])
     cY = int(M["m01"] / M["m00"])
   else:
     cX,cY = 0,0

   cv2.drawContours(image, [cnts], -1, (0, 255, 0), 2)
   cv2.circle(image, (cX, cY), 8, (255, 255, 255), -1)
   cv2.imwrite("imagee.jpg",image)
   cntrs,hierarchy      =      cv2.findContours(image,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
for cnts in cntrs:
   contour = np.concatenate(cntrs)
   x, y, w, h = cv2.boundingRect(contour)
   global newRoi
   newRoi = image[y:y+h, x:x+w]
   cv2.imwrite("cntt.jpg", newRoi)

   inv = cv2.bitwise_not(newRoi) kontur =
   inv.copy() cv2.imwrite('kontur.jpg',
   kontur)
```

Modul program 1.3. Proses normalisasi contour citra

Selanjutnya dilakukan tahap segmentasi citra untuk memisahkan tiap karakter citra. Tahap ini diawali dengan proses erosi kontur citra dengan kernel yang berukuran 100,7 dan dari proses erosi tersebut, akan menghasilkan citra dengan beberapa *bar* berwarna hitam dan putih. Kemudian mendeteksi tepian dari citra erosi tersebut dengan menggunakan deteksi tepi Canny dan dilanjutkan dengan hough line transform untuk mendeteksi garis yang terbentuk dari citra deteksi tepi. Dari fungsi tersebut titik- titik yang didapatkan kemudian dijadikan 1 menjadi *list* dan ditambahkan titik dari garis paling ujung yaitu lebar,tinggi dan lebar, nol. *List* tersebut kemudian diurutkan berdasarkan koordinat x dan dilanjutkan dengan menggambar garis dari koordinat-koordinat tersebut pada citra kontur. Citra kontur yang sudah bergaris disimpan ke dalam folder. Kemudian melakukan pemisahan untuk setiap karakter berdasarkan koordinat dari garis-garis yang terbentuk.

Selanjutnya dilakukan proses *cropping* sesuai dengan kotak yang telah dibentuk tadi. Citra yang terpotong memiliki ukuran yang tidak sama, oleh karena itu dilakukan penambahan bingkai putih untuk membuat citra menjadi berukuran persegi. Acuan ukuran yang dipakai berdasarakan tinggi dari citra *cropping*. Kemudian citra disimpan dengan nama resize_{}.jpg, tanda {} merupakan indeks x iterasi dari perulangan.

```
cntr = cv2.imread('kontur.jpg')
kernelerosi = np.ones((100,7), np.uint8)
erosi2 = cv2.erode(cntr, kernelerosi, iterations = 1)
cv2.imwrite('erosi2.jpg', erosi2)
deteksisegmen =
cv2.Canny(erosi2,25,255,L2gradient=False)
cv2.imwrite('deteksi_segmen.jpg', deteksisegmen)


m=[]
minLineLength = 100
maxLineGap = 10
lines = cv2.HoughLinesP(deteksisegmen, 1, np.pi/180,
15, minLineLength, maxLineGap)
for x in range(0, len(lines)):
       for x1,y1,x2,y2 in lines[x]:
            m.append(((x1,y1),(x2,y2)))
             height = y1
             width = inv.shape[1]
             m.append(((width,height), (width,0)))


sorted_m=sorted(m, key=lambda x: x[0][0])
for i in range(len(sorted_m)):

cv2.line(kontur,sorted_m[i][0],sorted_m[i][1],(0,0,255
),1)
cv2.imwrite('hough1.jpg',kontur)
h=0
v=0
ujungy=0
ujung= len(inv)
for x in range(0, len(sorted_m)):
            for z,y in (sorted_m[x]):
                if (y!=0):
                    enam = cv2.rectangle(newRoi, (h,
v), (z,y), (0,0,255))
```

```
                    putih2 = cv2.rectangle(kontur, (h,
v), (z,y), (255,255,255))
                    crop = putih2[0:y, h:z]
                    size = crop.shape[:2]
                    max_dim = max(size)
                    delta_w = max_dim - size[1]
                    delta_h = max_dim - size[0]
                    top, bottom = delta_h//2, delta_h-
(delta_h//2)
                    left, right = delta_w//2, delta_w-
(delta_w//2)
                    color = [255, 255, 255]
                    image_out =
cv2.copyMakeBorder(crop, top, bottom, left, right,
cv2.BORDER_CONSTANT, value=color)

cv2.imwrite('resize_{}.jpg'.format(x), image_out)
                    h =z
                    ujungy = y
```

Modul program 1.4. Segmentasi citra

Proses yang terakhir yaitu proses identifikasi citra. Pada proses ini, citra melewati beberapa proses yaitu proses ekstraksi fitur menggunakan Gabor Wavelet yang berfungsi untuk mendapatkan beberapa ciri yang dibutuhkan dari setiap citra untuk membedakan citra huruf satu dengan yang lainnya. Hasil ekstraksi ciri akan disimpan didalam uji.csv. Fungsi untuk ekstraksi ciri Gabor Wavelet akan dijelaskan lebih lanjut dalam beberapa modul program berikut.

```
stds = []
meann = []
varians = []
med = []
l_med = []
l_std = []
l_meann = []
l_var = []
psi =0.5
gamma =0.5
sigma =10
scale = [3,6,13,28,58]
sudut = [0,30,45,60,90,120,135,180]
```

Modul program 1.5. Proses Inisiasi

Dalam modul program tersebut berisikan inisialisasi variabel yang diperlukan dalam proses ekstraksi ciri. Variabel tersebut terdiri dari variabel yang berfungsi untuk menyimpan nilai hasil dari ekstraksi ciri dan variabel yang membawa nilai sebagai parameter dalam membuat kernel Gabor Wavelet.

```
      for j in range(0 , banyak_char, 1) : pathfile=
            'resize_{}.jpg'.format(j)gambar =
            cv2.imread(pathfile)
            dsize = (60,60)
            src = cv2.resize(gambar, dsize)src =
            img_as_float(src)
            for lamda in scale:
                for theta in sudut:

                        kernel                                  =
gabor(src,sigma,theta,lamda,psi,gamma)
                        rata, var, std, median =kernelder =
                        theta
                        label_median     =     'Median
'+str(der) +','+str(lamda)
                        label_std  =  'Std  '+str(der)
+','+str(lamda)
                        label_mean = 'Mean  '+str(der)
+','+str(lamda)
                        label_var = 'Varian '+str(der)
+','+str(lamda)

                        l_std.append(label_std)
                        l_meann.append(label_mean)
                        l_var.append(label_var)
                        l_med.append(label_median)

                        stds.append(std)
                        meann.append(rata)
                        varians.append(var)
                        med.append(median)
            m =dict(zip(l_meann,meann))e
            =dict(zip(l_med,med))
            v =dict(zip(l_var,varians))s
            =dict(zip(l_std,stds)) merge ={**s,
            **m, **v, **e}
            df = pd.DataFrame.from_dict([merge])df['Nama
            File'] = pathfile
            if (j==0):
                df.to_csv("uji.csv", sep = ',',
index=False)
            else:
                df.to_csv("uji.csv", sep = ',',
index=False, mode = 'a', header=False)
```

Modul program 1.6 Proses Ekstraksi ciri

Pada modul program tersebut melakukan proses pemanggilan citra tiap huruf dengan menggunakan perulangan sebanyak karakter yang disimpan sebelumnya. Citra huruf tersebut kemudian diubah ukurannya menjadi 60x60 dan mengkonversi citra menjadi nilai *float.* Setelah itu perulangan untuk setiap

lamda dan theta melakukan pemanggilan fungsi gabor dengan mengirimkan parameter berupa citra huruf, sigma, theta, lamda, psi dan gamma. Dari fungsi tersebut akan dihasilkan nilai ciri berupa mean, variansi, standar deviasi dan median. Nilai tersebut kemudian disimpan ke dalam list telah disiapkan.

Setelah ektraksi ciri pada 1 citra selesai, *list* tersebut akan digabungkan dengan label nama ciri menjadi *dictionary.* Terdapat 4 *dictionary* yaitu standar deviasi, mean, variansi dan median yang kemudian dilakukan penggabungan. Setelah itu akan disimpan ke dalam uji.csv.

```
def gabor(img,sigma, theta, Lambda, psi, gamma):"""Gabor
            feature extraction."""" sigma_x = sigma
            sigma_y = float(sigma) / gammaF =
            np.sqrt(2)/2**Lambda

            (y, x) = np.meshgrid(np.arange(-30,30),
np.arange(-30,30))

            # Rotation
            x_theta = x * np.cos(theta) + y *
np.sin(theta)
            y_theta = -x * np.sin(theta) + y *
np.cos(theta)
            #real gaborreal_gb =
(1/(2*np.pi*sigma_x*sigma_y))*np.exp(-.5 * (x_theta **
2 / sigma_x ** 2 + y_theta ** 2 / sigma_y ** 2)) *np.cos(2 *
np.pi * F * x_theta + psi)
            im_gb = (1/(2*np.pi*sigma_x*sigma_y))*np.exp(-.5 *
(x_theta **
2 / sigma_x ** 2 + y_theta ** 2 / sigma_y ** 2)) *np.sin(2 *
np.pi * F * y_theta + psi)

            img_real = cv2.filter2D(img, cv2.CV_32F,
real_gb)

            img_im = cv2.filter2D(img,cv2.CV_32F,
im_gb)

            lblgabor = str(theta) +','+str(Lambda)

cv2.imwrite('gabor{}.jpg'.format(lblgabor), real_gb)
            mag = np.sqrt((img_real**2)+(img_im**2))mag =

            abs(mag)/np.max(mag)

            rata = np.mean(mag) varian =
            np.var(mag)
            std = np.std(mag, ddof=1)median =
            np.median(mag)
            return  rata, varian, std, median
```

Modul program 1.7 Proses Gabor Wavelet

Modul program 1.7. merupakan proses dalam mengimplementasikan Gabor Wavelet. Diawali dengan membuat kernel riil dan kernel imajiner dengan menggunakan rumus gabor dan parameter yang dikirimkan dari kelas utama. Setelah kernel terbentuk, untuk setiap kernel riil dan imajiner dilakukan proses konvolusi dengan citra huruf yang dikirimkan. Dari konvolusi tersebut dihasilkan citra riil dan citra imajiner, untuk menggabungkannya menjadi citra magnitude. Dari

citra magnitude akan dilakukan proses perhitungan nilai ciri yang dibutuhkan. Untuk rata-rata melalui rata = np.mean(mag), untuk variansi melalui varian = np.var(mag), kemudian untuk standar deviasi melalui std = np.std(mag, ddof=1) dan untuk nilai median dengan median = np.median(mag).

Selanjutnya melakukan proses klasifikasi menggunakan SVM untuk mengidentifikasi karakter braille.

```
                'Std 120,58', 'Mean 120,58', 'Varian
120,58','Median 120,58',
                'Std 135,58' , 'Mean 135,58', 'Varian
135,58' ,'Median 135,58',
                'Std 180,58' , 'Mean 180,58', 'Varian
180,58', 'Median 180,58']].values
    clf = OneVsRestClassifier(SVC()).fit(X, y)def
    listToString(s):
                    # initialize an empty stringstr1
                    = ""
                    # delete space
                      for x in range (len(s)):
                          if(s[x] == " "):
                              s[x] =""
                    # traverse in the stringif
                    (s[0] == "#"):
                        for ele in range (1,len(s)):if
                            (s[ele] == "a"):
                                str1 += "1"
                            if (s[ele] == "b"):str1
                                += "2"
                            if (s[ele] == "c"):str1
                                += "3"
                            if (s[ele] == "d"):str1
                                += "4"
                            if (s[ele] == "e"):str1
                                += "5"
                            if (s[ele] == "f"):str1
                                += "6"
                            if (s[ele] == "g"):str1
                                += "7"
                            if (s[ele] == "h"):str1
                                += "8"
                            if (s[ele] == "i"):str1
                                += "9"
                            if (s[ele] == "j"):str1
                                += "0"
                    elif (s[0]== "*"):
                        if(s[2] == "*"):
                            for ele in range
(3,len(s)):
                                str1 += s[ele].upper()
                        else:
                            for ele in range
(1,len(s)):
                                str1 += s[ele]
                            str1 = str1.capitalize()
                    else:
                        for ele in s:
                            str1 += ele
                    print("input gtts"+str1)#
                    return string
                    return str1s=
   clf.predict(test)
   kata = listToString(s)
   forsuara = kata

   return kata,forsuara
```

Modul program 1.8. Modul klasifikasi SVM

Modul program 1.8 merupakan pseudocode proses klasifikasi. Proses klasifikasi diawali dengan memanggil fungsi klasifikasi. Dalam fungsi klasifikasi tersebut melakukan pemanggilan data latih dengan nama file Gaborlatih1crop.csv dan data uji dengan nama file uji.csv. Pada masing-masing data latih dan data uji akan diambil nilai cirinya secara manual berdasarkan *header* pada *file* csv tersebut. Kemudian mendefinisikan kelasnya bersarkan nama kelas pada kolom huruf di data latih. Daftar nilai tersebut kemudian diubah menjadi 1xn array. Kemudian dengan menggunakan fungsi OneVsRestClassifier library sklearn sebagai metode klasifikasi pada SVM multikelas. Setelah itu melakukan prediksi untuk setiap data pada data uji melalui s = clf.predict(test) dari hasil prediksi tersebut terdapat karakter spasi, untuk menghilangkannya dengan memanggil kelas listToString melalui kata = (listToString(s)). Pada fungsi ini, setelah karakter spasi dihilangkan, akan dilakukan pengecekan karakter, apakah terdapat penanda angka ataupun kapital, jika tidak ada maka akan ditampilkan string dari kelas abjad dan tanda baca. Selanjutnya jika terdapat penanda angka maka karakter berikutnya yang berupa huruf a-j akan diubah menjadi angka 1,2,3,4 sampai dengan 0. Kemudian apabila ditemukan penanda kapital maka dilanjutkan dengan pengecekan karakter setelahnya apakah penanda kapital kembali atau huruf abjad. Jika huruf abjad, maka hanya huruf pertama saja yang berupa huruf kapital, namun jika bertemu penanda kapital lagi maka semua karakter

merupakan huruf kapital semua.

```
def suara(forsuara):
     mytext = forsuara
     language = 'id'
     myobj    =    gTTS(text=mytext,    lang=language,
slow=False)

     # Saving the converted audio in a mp3 file named
welcome
     file = "welcome.mp3"
     myobj.save(file)
```

Modul program 1.9. Proses string menjadi suara

Pada modul program tersebut, string yang telah diproses dan didapatkan dari hasil klasifikasi akan dikirimkan pada fungsi suara dengan variabel parameter forsuara. Kemudian menentukan bahasa yang akan digunakan untuk membaca string tersebut. Dengan menggunakan fungsi gTTs yang ada pada library gtts akan melakukan proses pembacaan string menjadi suara sebagai hasil keluarannya. Dalam fungsi tersebut terdapat 3 variabel parameter, yaitu variabel text yang berisikan string yang akan dibaca, lang yang merupakan bahasa yang digunakan dan variabel slow yang bernilai True atau False yang mempengaruhi kecepatan dalam pengucapan kata. Kemudian, *output* suara tersebut akan disimpan sebagai welcome.mp3.

```
k_svm, forsuara = klasifikasi()
self.textBrowser.setText(k_svm) df1 =
pd.read_csv("uji.csv") model =
DataFrameModel(df1)
self.tableView.setModel(model)
self.pushButton.setEnabled(True)
suara(forsuara)
self.btnsuara.setEnabled(True)
```

Modul program 1.10. Proses pemanggilan fungsi

Diawali dengan pemanggilan fungsi klasifikasi yang hasilnya akan disimpan pada variabel k_svm dan forsuara secara berurutan. Setelah itu untuk hasil identifikasi yang ada pada variabel k_svm akan ditulis pada *text browser* sebagai hasil identifikasi melalui self.textBrowser.setText(k_svm). Kemudian, pada *TableView* akan ditampilkan tabel yang berisikan hasil ekstraksi ciri data uji.

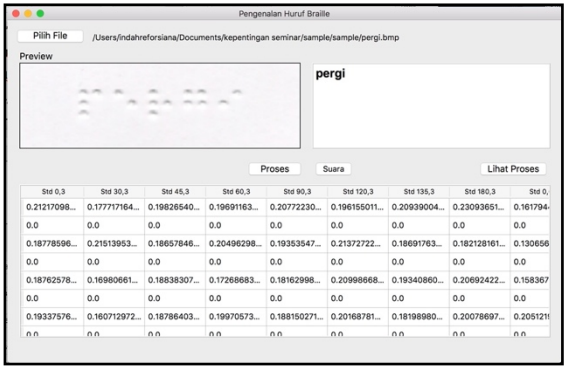
Pada pemanggilan fungsi suara akan dikirimkan variabel forsuara yang berisikan string hasil identifikasi, kemudian jika proses pada fungsi suara telah selesai, maka tombol suara akan diaktifkan.

```
def suara_clicked(self):
  playsound("welcome.mp3")
```

Modul program 1.11. Proses tombol suara

Pada saat tombol suara ditekan, maka akan melakukan proses pada fungsi suara_clicked. Dalam fungsi tersebut menggunakan fungsi *playsound* pada library *playsound* untuk memutar suara pada file welcome.mp3 yang telah disimpan sebelumnya.

Tampilan aplikasi setelah pengolahan citra dan diolah menggunakan *Gabor Wavelet* untuk mendapatkan ciri pada masing-masing gambar untuk diklasifikasikan menggunakan *Support Vector Machine* agar dapat teridentifikasi, terdapat pada gambar berikut.



Gambar 1.5 Tampilan Hasil Identifikasi

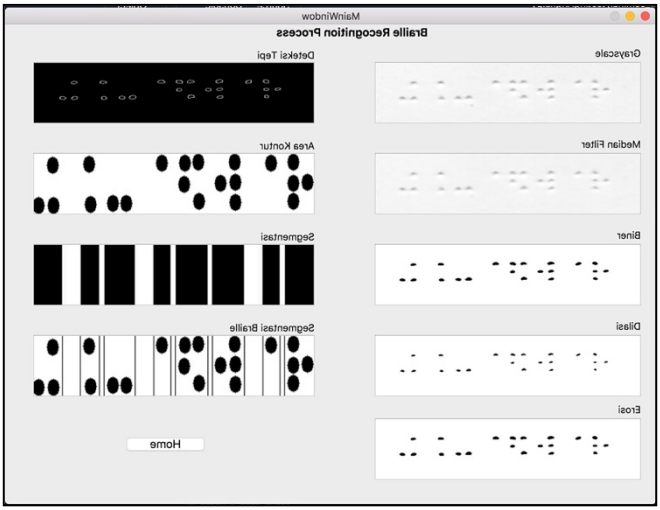
2.4. Tampilan Hasil Pengolahan Citra

Halaman lihat proses merupakan halaman yang berfungsi untuk menampilkan hasil dari pengolahan citra braille. Halaman ini akan memanggil citra-citra yang dihasilkan dan *user* dapat melihat citra yang dihasilkan dalam proses pengolahan citra. Untuk menampilkan halaman lihat proses, maka *user* perlu menekan tombol lihat proses. Berikut modul program pada saat tombol lihat proses ditekan.

```
def lihatproses_clicked(self):
        self.ui = Ui_second()
        self.ui.show()
```

Modul program 1.12. Lihat tombol proses

Setelah tombol lihat proses ditekan maka akan tampil halaman lihat proses yang berisikan citra *grayscale, median filter,* biner, dilasi, erosi, deteksi tepi, area kontur, segmentasi citra, dan segmentasi braille seperti pada gambar berikut.



Gambar 1.6 Tampilan Lihat Proses

Pada halaman lihat proses, berikut fungsi untuk menampilkan halaman lihat proses terdapat pada modul program berikut.

```
class Ui_second(QMainWindow):
    def_init_(self, parent=None):
        super()._init_(parent)
        self.ui = uic.loadUi('../second page.ui', self)

        pgray = Image.open('grayIMG.jpg')
        qgray= ImageQt(pgray)
        pixmap=QtGui.QPixmap.fromImage(qgray)
self.gray.setPixmap(pixmap.scaled(351,81,transformMode
=QtCore.Qt.SmoothTransformation))
        pfilter = Image.open('filterMed.jpg')
        qfilter= ImageQt(pfilter)
        pixmap=QtGui.QPixmap.fromImage(qfilter)
self.median.setPixmap(pixmap.scaled(351,81,transformMo
de=QtCore.Qt.SmoothTransformation))

        pbiner = Image.open('binerIMG.jpg')
        qbiner= ImageQt(pbiner)
        pixmap=QtGui.QPixmap.fromImage(qbiner)
self.biner.setPixmap(pixmap.scaled(351,81,transformMod
e=QtCore.Qt.SmoothTransformation))

        pdilasi = Image.open('dilImg.jpg')
        qdilasi= ImageQt(pdilasi)
```

```
        pixmap=QtGui.QPixmap.fromImage(qdilasi)
self.dilasi.setPixmap(pixmap.scaled(351,81,transformMo
de=QtCore.Qt.SmoothTransformation))

        perosi = Image.open('erosiImg.jpg')
        qerosi= ImageQt(perosi)
        pixmap=QtGui.QPixmap.fromImage(qerosi)
self.erosi.setPixmap(pixmap.scaled(351,81,transformMod
e=QtCore.Qt.SmoothTransformation))

        pcanny = Image.open('detCanny.jpg')
        qcanny= ImageQt(pcanny)
        pixmap=QtGui.QPixmap.fromImage(qcanny)
self.canny.setPixmap(pixmap.scaled(371,81,transformMod
e=QtCore.Qt.SmoothTransformation))

        pkontur = Image.open('kontur.jpg')
        qkontur= ImageQt(pkontur)
        pixmap=QtGui.QPixmap.fromImage(qkontur)
self.kontur.setPixmap(pixmap.scaled(371,81,transformMo
de=QtCore.Qt.SmoothTransformation))

        psegmen = Image.open('erosi2.jpg')
        qsegmen= ImageQt(psegmen)
        pixmap=QtGui.QPixmap.fromImage(qsegmen)
self.segmen.setPixmap(pixmap.scaled(371,81,transformMo
de=QtCore.Qt.SmoothTransformation))

        pbraille = Image.open('hough1.jpg')
        qbraille= ImageQt(pbraille)
        pixmap=QtGui.QPixmap.fromImage(qbraille)
self.houghline.setPixmap(pixmap.scaled(371,81,transfor
mMode=QtCore.Qt.SmoothTransformation))

        self.pushButton.clicked.connect(self.reject)
    def reject(self):
        self.close()
```

Modul program 1.13. Proses tampilan halaman

Pada halaman lihat proses, *user* dapat lebih memahami bagaimana proses atau apa saja proses yang perlu dilakukan dari citra huruf braille untuk dapat mendapatkan hasil identifkasi berupa huruf abjad. Untuk kembali melakukan identifikasi citra yang baru, maka *user* dapat menekan tombol *home.*

Dalam tahap mengimplementasikan *Gabor Wavelet* sebagai pengekstraksi ciri pada citra yang memiliki

tekstur, dibutuhkan beberapa parameter diantaranya adalah panjang gelombang(lamda), sudut orientasi(theta), sigma, dan gamma. Perbedaan jumlah nilai pada setiap parameter yang digunakan dalam mengekstraksi fitur akan sangat mempengaruhi tingkat akurasi dalam proses identifikasi. Misalkan untuk nilai parameter lamba yang digunakan hanya 3 buah saja akan memiliki tingkat akurasi yang lebih rendah dibandingkan dengan menggunakan nilai parameter lamda sebanyak 5 buah. Sehingga, semakin banyak nilai parameter yang digunakan dalam mengekstraksi ciri suatu citra dengan menggunakan metode *Gabor Wavelet*, maka semakin tinggi pula tingkat akurasi yang didapatkan.

Kesalahan dalam mengidentifikasi huruf dikarenakan nilai hasil dari ekstraksi ciri pada citra tersebut pada proses klasifikasi cenderung memiliki kedekatan dengan kelas lain, dibandingkan dengan kelas huruf yang sebenarnya. Selain itu, kemiringan dalam penulisan huruf braille dan juga adanya titik atau noda pada kertas yang digunakan untuk menulis huruf braille dapat mempengaruhi hasil identifikasi. Sebagai contoh, terdapat noda berupa garis yang berada dekat dengan titik braille yang akan diolah. Pada hasil pembacaan tetap dapat dihasilkan huruf sesuai dengan huruf yang nilai cirinya mendekati sesuai dengan data latih sebelumnya.

PENUTUP

Latihan Soal

1. Apa itu huruf braille ?
2. Apa nama alat yang digunakan untuk membuat huruf braille ?
3. Bagaimana cara seorang tunanetra dalam membaca huruf braille?
4. Apa yang dimaksud dengan ekstraksi ciri ?
5. Apa tujuan dari dilakukannya ekstraksi ciri?
6. Jelaskan apa itu Gabor Wavelet ?
7. Sebutkan kelebihan dari metode Gabor Wavelet ?
8. Penelitian apa saja yang dapat menerapkan metode Gabor Wavelet ?
9. Apa saja parameter yang digunakan untuk membangun kernel Gabor Wavelet?
10. Dalam persamaan Gabor Wavelet, apa yang membedakan dari kernel imajiner dan kernel riil ?
11. Ciri apa saja yang dapat diekstrak dengan Gabor Wavelet ?
12. Apa itu vektor ciri ?
13. Apa yang dimaksud dari Support Vector Machine(SVM) ?
14. Apa yang dimaksud dengan margin?
15. Apa tujuan dari proses pembelajaran pada SVM ?
16. Apa itu *empirical risk* ?
17. Sebutkan karakteristik SVM secara umum menurut Nugroho(2003)!

18. Apa itu kernel trick ?
19. Sebutkan kernel yang umum digunakan dalam SVM ?
20. Apa itu metode sekuential training ?
21. Sebutkan apa saja metode yang dapat digunakan pada klasifikasi multikelas ?
22. Jelaskan tahapan proses dalam sekuential training!

DAFTAR PUSTAKA

Adak, C. (2014). Gabor Filter and Rough Clustering Based Edge Detection. *International Conference on Human Computer Interactions (ICHCI).* Chennai, India: IEEE.

Boswell, D. (2002, Agustus). Introduction to Suppot Vector Machines.

Cortes, Vapnik, C., & Vladimir.(1995). Support-Vector Networks. *Machine Learning* **20,** 273–297

Harimi, A. (2018). Strategi Pembelajaran Kemahiran Menulis Bagi Peserta Didik Tunanetra. *Prosiding Konverensi Nasional Bahasa Arab IV*, (pp. 51-56).

Hsu, C. W., & Lin, C. J. (2002). A Comparison of Methods for Multiclass Support Vector Machines. *IEEE TRANSACTIONS ON NEURAL NETWORKS, Vol 13, No 2.*

Jayanti, N., Novianti, K., & Sumalya, I. (2017). Implementasi Metode Support Vector Machine Pada Sistem Pengenalan Jejaitan. *Seminar Nasional Teknologi Informasi dan Multimedia 2017*, 163-168.

Nugroho, A. W. (2003). Support Vector Machine Teori dan Aplikasinya dalam Bioinformatika.

Rangayyan, R. M. (2007). A Review of Computer-Aided Diagnosis of Breast Cancer: Toward The Detection of Subtle Signs. *Journal of The Franklin Institute*, 312-348.

Santosa, B. (2007). *Data Mining Teknik Pemanfaatan Keperluan Bisnis.* Yogyakarta: Graha Ilmu.

Sembiring, K. (2007). Penerapan Teknik Support Vector Machine untuk Pendeteksian Instruksi pada Jaringan. *ITB, Bandung*.

Tarsidi, D. (2000, Mei). Pedoman Format Braille. *Seminar Nasional tentang Produksi Braille*.

Wisesty, U., & Mutiah, T. (2016). Implementasi Gabor Wavelet dan Support Vector Machine pada Deteksi Polycystic Ovary (PCO) Berdasarkan Citra Ultrasonografi. *Indonesian Journal of Computing*, 67-82.

Yong-zhao, Z., Jing-fu, Y., De-jiao, N., & Peng, C. (2004). Facial Expression Recognition Based on Gabor Wavelet Transformation and Elastic Templates Matching. *IEEE Computer Society*.

Penerbit :
LPPM
UPN Veteran Yogyakarta

Lembaga Penelitian & Pengabdian Masyarakat
Universitas Pembangunan Nasional "Veteran" Yogyakarta

ISBN 978-623-6896-74-7