Verifikasi Tanda Tangan

Menggunakan Teknik Pemrosesan Gambar

Muhammad Iqbal Sigid 13519152
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): sigid.iqbal123@gmail.com

***Abstak*—Tanda tangan adalah mekanisme yang utama untuk otentikasi dan otorisasi, seperti pada dokumen atau transaksi legal. Karena tanda tangan adalah sebuah tulisan tangan yang berbeda dengan *password*, verifikasi otomatis memerlukan pemrosesan citra. Pada makalah ini dibuat sebuah program verifikasi tanda tangan yang menggunakan teknik pemrosesan citra, dimulai dari *image preprocessing* dengan melakukan *cropping, filtering, binarization,* dan *thinning* kemudian *feature extraction* yang terdiri dari fitur *Normalised signature area, aspect ratio, centroid,* dan *endpoint*. Dengan menggunakan empat fitur tersebut program akan menentukan *validity* dari tanda tangan. Kinerja dari program sudah cukup baik, tetapi masih dapat dikembangkan lagi.**

Kata Kunci—Signature, Pemrosesan Citra, MATLAB

I. PENDAHULUAN

Tanda tangan adalah sebuah gambaran nama atau identitas seseorang yang digunakan sebagai bukti identitas, misalnya pada sebuah dokumen. Tanda tangan biasanya ditulis tangan dan umumnya unik untuk seseorang, walaupun tidak memungkinkan bahwa dua orang memiliki tanda tangan yang mirip. Penggunaan tanda tangan sebagai bukti identitas ini menjadikan verifikasi tangan tangan perlu untuk dilakukan.

Verifikasi tanda tangan idealnya dapat mengkonsiderasikan teknik penggambaran tanda tangan tersebut, seperti bagaimana urutan penggambaran garis dan kecepatan penggambaran. Namun, pada makalah ini verifikasi tanda tangan hanya dilakukan pada gambar tanda tangan itu sendiri. Untuk melakukan verifikasi tanda tangan ini, akan dilakukan beberapa teknik pemrosesan citra untuk membersihkan dan mengekstraksi fitur gambar yang kemudian akan diklasifikasikan fitur yang telah diperoleh tersebut.

II. LANDASAN TEORI

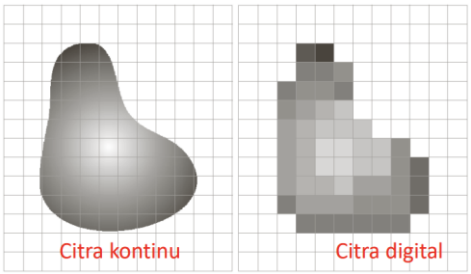
A. *Citra Digital*

Citra merupakan sinyal kontinu yang hanya bisa dilihat dari satu arah saja yaitu $f(x, y)$, dengan $f(x, y)$ menyatakan intensitas cahaya pada posisi (x, y). Agar citra dapat diolah oleh komputer digital, maka citra perlu didigitalisasikan menjadi citra digital.

Citra digital adalah representasi citra kontinu melalui penyuplikan (sampling) secara ruang dan waktu. Proses digitalisasi citra terbagi atas dua tahap, yaitu :

1. Penyerokan yaitu digitalisasi secara spasial (x, y) bertujuan untuk menentukan seberapa banyak pixel yang diperlukan untuk merepresentasikan citra kontinu, dan bagaimana pengaturannya.
2. Kuantisasi yaitu mengangkakan nilai intensitas $f(x, y)$ menjadi integer. Bertujuan untuk memetakan nilai dari sinyal kontinu menjadi G buah nilai diskrit (G buah level).

Dari kedua tahap tersebut, dapat dihasilkan sebuah citra digital yang berbentuk matriks berisi informasi warna dari citra asli. Gambar II.1 adalah ilustrasi contoh hasil digitalisasi pada sebuah citra.



Gambar II.1 Proses digitasi citra

(Sumber: Rinaldi Munir)

B. *Citra Biner*

Citra tanda tangan hanyalah sebatas coretan unik pada kertas sehingga citra dapat direpresentasikan dengan citra biner. Citra biner adalah citra yang hanya menggunakan dua warna untuk merepresentasikan gambar. Warna yang digunakan adalah hitam dan putih dengan digit 0 merepresentasikan warna hitam dan digit 1 untuk warna putih. Citra biner digunakan untuk merepresentasikan tanda tangan karena fitur yang diperlukan untuk pengenalan tanda tangan hanyalah bentuk atau polanya. Kelebihan dari citra biner adalah kontras yang tinggi dan memerlukan sedikit memori untuk penyimpanan.

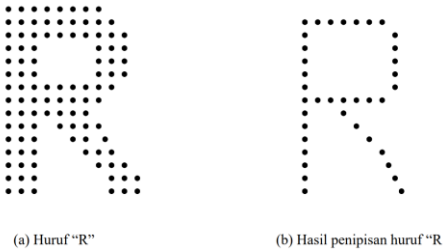
Gambar II.2. Contoh citra biner berupa *barcode* dan *QR code*

(Sumber: Rinaldi Munir)

Untuk mendapatkan citra biner dapat dilakukan *thresholding* atau pengambangan. Karena tanda tangan biasanya digambar dengan tinta hitam pada kerta berwarna terang, pengambangan dapat dilakukan misalnya pada nilai pixel 30. Untuk pixel kurang dari atau sama dengan nilai ambang, maka akan dijadikan 0 dan pixel yang bernilai lebih akan dijadikan 1. Jika citra merupakah RGB, citra dapat diubah terlebih dahulu menjadi citra *grayscale*.

C. *Image Thinning*

Thinning atau penipisan gambar adalah suatuoperasi pemrosesan citra biner untuk mereduksi citra menjadi rangkanya saja (*skeleton*). Tujuan dari penipisan citra adalah untuk membuang bagian citra yang *redundant* atau tidak begitu diperlukan dan menyisakan bagian citra yang penting saja. Penipisan citra dapat berguna untuk *pattern recognition* karena berbagai gambar dapat memiliki pola yang sama, tetapi ketebalannya saja yang berbeda. Penipisan citra dapat berguna agar komputer dapat lebih mudah untuk merekognisi pola tersebut.



Gambar II.3. Contoh proses penipisan citra huruf 'R'

(Sumber: Rinaldi Munir)

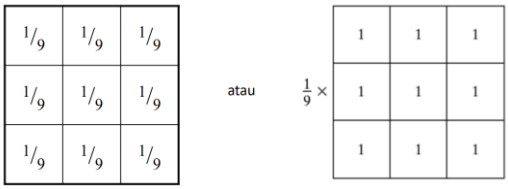
Secara umum, algoritma penipisan citra akan memeriksa semua pixel. Jika pixel tersebut bukan pixel yang menghubungkan dua pixel dan bukan pixel ujung, maka pixel tersebut akan dibuang. Untuk melakukan penipisan citra terdapat dua syarat yang harus dipenuhi dalam membuang pixel yang redundan, yaitu:

1. Mempertahankan keterhubungan pixel-pixel objek. Dengan kata lain, tidak menyebabkan bentuk objek menjadi terputus.
2. Tidak memperpendek ujung lengan dari bentuk yang ditipiskan.

D. *Image Filtering*

Penapisan citra atau *image filtering* adalah memodifikasi citra berdasarkan nilai pixel tetangganya. Penapisan sering digunakan untuk deteksi tepi citra, *blurring* atau *smoothing*, dan *sharpening*. Operasi yang dilakukan untuk penapisan citra adalah operasi konvolusi dengan penapis tertentu sesuai dengan kebutuhan.

Salah satu contoh penapis yang digunakan untuk *image smoothing* adalah penapis rerata (*median filter*). Penapis rerata adalah sebuah matriks dengan ukuran tertentu dengan nilai yang sama pada setiap baris dan kolom. Contoh penapis rerata dapat dilihat pada Gambar II.4. Ukuran dan nilai pada matriks akan mempengaruhi *smoothness* dari hasil citra.



Gambar II.4 Contoh penapis rerata

(Sumber: Rinaldi Munir)

E. *Image Transformation*

Transformasi citra adalah perubahan domain pada citra, atau memetakan citra ke ruang vektor lain. Transformasi citra adalah sebuah operasi geometrik yang dapat direpresentasikan dalam sebuah matriks. Contoh transformasi citra antara lain adalah translasi dan rotasi.

Translasi adalah pergeseran suatu titik ke titik yang baru secara linear. Translasi dapat dilakukan dengan melakukan operasi penjumlahan matriks. Gambar II.5 menunjukkan matriks translasi.

$$\mathbf{x} = \mathbf{u} + \mathbf{t}, \quad \mathbf{u} = \mathbf{x} - \mathbf{t}$$
where
$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Gambar II.5 Matriks translasi

Selain translasi, terdapat juga transformasi rotasi. Rotasi adalah perputaran titik ke titik baru berdasarkan suatu *point of reference* atau titik referensi. Titik ini umumnya dilakukan terhadap titik (0,0) atau titik tengah dari citra. Matriks rotasi direpresentasikan pada Gambar II.6 berikut.

$$\mathbf{x} = \mathbf{R}\mathbf{u}, \quad \mathbf{u} = \mathbf{R}^T\mathbf{x}$$
where
$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Gambar II.6 Matriks rotasi

III. Implementasi Solusi

Implementasi program verifikasi tanda tangan digunakan program MATLAB. Terdapat dua tahapan dalam proses verifikasi ini, yaitu tahap *proprecessing* citra dan tahap *feature extraction*. Pada tahap *preprocessing*, citra akan diproses melalui beberapa operasi citra sehingga fitur citra dapat diekstraksi dengan baik, sedangkan tahap *feature extraction* akan dilakukan ekstraksi fitur agar kedua citra dapat dibandingkan.

Tahap *preprocessing* adalah sebagai berikut:

1. *Resizing*, untuk menyamakan ukuran citra
2. *Filtering*, untuk mereduksi *noise* yang ada pada citra jika ada
3. Konversi citra menjadi citra biner
4. *Thinning*, untuk mengurangi ketebalan dari tanda tangan
5. *Cropping*, untuk mengurangi bagian pinggir dari citra yang hanya berisi latar

Kemudian tahap *feature extraction* adalah sebagai berikut:

1. *Normalised signature area*, total area yang memiliki pixel berwarna hitam dibagi dengan jumlah pixel keseluruhan citra
2. *Aspect ratio*, rasio dari ukuran gambar, lebar/tinggi
3. *Signature centroid*, titik tengah dari tanda tangan tanpa memperhitungkan latar
4. *Signature endpoints*, jumlah titik ujung dari tanda tangan.

Fitur yang diekstraksi akan disimpan pada suatu matriks. Untuk menentukan kemiripan dari tanda tangan dengan referensinya, akan dihitung perbedaan pada empat fitur tersebut terhadap referensi. Kemudian untuk menentukan apakah tanda tangan adalah sama atau tidak, akan ada sebuah *threshold* dari nilai error fitur.

A. *Kode Program*

1. *Preprocessing*

```
% resize image
I2=imresize(I,[512 ,512]);

% convert to grayscale
I3=rgb2gray(I2);

% noise reduction
I3=medfilt2(I3);

% convert to binary
I3=im2double(I3);
I3=im2bw(I3);

% thinning image then reverse black & white
pixel
I3 = bwmorph(~I3, 'thin', inf);
I3=~I3;
im1 = I3;


% removing extra white space in sides
xstart=512;
xend=1;
ystart=512;
yend=1;
for r=1:512
    for c=1:512
        if((I3(r,c)==0))
            if (r<ystart)
                ystart=r;
            end
            if((r>yend))
                yend=r;
            end
            if (c<xstart)
                xstart=c;
            end
            if (c>xend)
                xend=c;
            end
       end
    end
end

%cutting the image and copying it to
another matrix
for i=ystart:yend
    for j=xstart:xend
        im((i-ystart+1),(j-
xstart+1))=I3(i,j);
    end
end
im_processed = im;
```

Kode di atas akan melakukan *preprocessing* pada citra. Operasi *resize*, *filtering*, *binarize*, dan *thinning* menggunakan fungsi yang telah disediakan oleh MATLAB. Untuk operasi *cropping* untuk membuang bagian tepi citra dilakukan dengan menghitung *xstart*, *ystart*, *xend*, *yend* dengan mencari pixel hitam terluar. Pixel yang berada di luar keempat pixel tersebut kemudian akan dibuang.

2. *Feature Extraction*

```
% Feature Extraction - NSA ----------------

PixelB = 0;
PixelA = 0;
for i=ystart:yend
    for j=xstart:xend
        if (im(i-ystart+1,j-xstart+1)== 0)
            PixelB = PixelB + 1;
        end
```

```
        PixelA = PixelA + 1;
    end
end

NSA = PixelB/PixelA;

% Feature Extration - Aspect Ratio --------

height_sign = yend-ystart;
length_sign = xend-xstart;
aspect_ratio = length_sign/height_sign

% Feature Extraction End Points -----------

i1 = im1;
[row, col, depth] = size(i1);
%add row%
addrow = ones(1, col);
i1 = [addrow; addrow; i1; addrow];
[row, col, depth] = size(i1);
%add column%
addcol = ones(row, 1);
i1 = horzcat(addcol, i1, addcol, addcol);
[row, col, depth] = size(i1);
i1=~i1;
crosspoints=0;
 for r = 3:row-1
        for c = 2:col-2
            if(i1(r,c)==1)
                if (i1(r-1,c-1)+i1(r-
1,c)+i1(r-1,c+1)+i1(r,c-
1)+i1(r,c+1)+i1(r+1,c-
1)+i1(r+1,c)+i1(r+1,c+1)==1)

crosspoints=crosspoints+1;
                    %disp(i1(r,c))
                end
            end
        end
 end

Feat_Val = [ NSA aspect_ratio crosspoints
centroid];
```

Kode di atas mengekstraksi tiga fitur yaitu NSA, *aspect ratio*, dan *end points*. NSA dihitung dengan membagi jumlah pixel hitam dengan jumlah seluruh pixel dan a*spect ratio* dihitung dengan membagi lebar dengan tinggi citra. Untuk *end points*, ditambahkan *padding* pada gambar kemudian dihitung jumlah pixel yang hanya memiliki satu tetangga.

Perhitungan sentroid dilakukan dengan membuat sebuah array yang menyimpan koordinat pixel hitam. Array tersebut kemudian dijumlahkan dan dibagi dengan lebar atau tinggi citra. Keempat fitur tersebut kemudian disimpan pada array 'Feat_Val'.

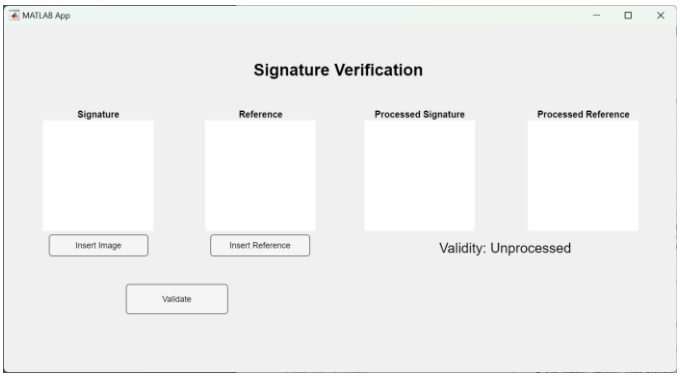
3. Perhitungan Error

```
Feat_Err = [ 0.006 0.8 6 [[0.0750 0.0750]
[0.0750 0.0750]] ]; % NSA aspect_ratio
crosspoints centroid

flag = 1;
for i=1:4
    if abs(feat_sign(i)-feat_ref(i)) >
Feat_Err(i)
        flag = 0;
        break;
    end
end
```

Fitur dari citra masukan dan citra referensi akan dihitung perbedaannya dan jika keempat fitur tersebut kurang dari *threshold*, maka hasil adalah valid.

B. Graphical User Interface



Gambar III.1 GUI dari program MATLAB yang dibuat

GUI dari program memiliki komponen sebagai berikut:

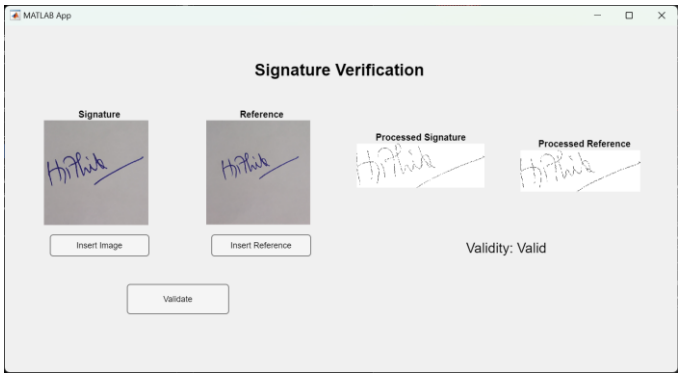
- Tampilan citra tanda tangan masukan dan tanda tangan referensi.
- Tampilan citra untuk tanda tangan masukan dan referensi hasil *preprocessing*.
- Tombol untuk memilih citra tanda tangan sebagai citra masukan dan referensi.
- Tombol untuk memulai validasi.
- Hasil validasi.

Pertama pengguna dapat memilih citra masukan dan referensi dengan menekan tombol yang tersedia. Setelah kedua citra terbaca oleh program, pengguna dapat menekan tombol validasi. Hasil citra yang telah di-*preprocess* akan ditampilkan dan hasil verifikasi akan muncul pada text di kanan bawah.

IV. HASIL DAN ANALISIS

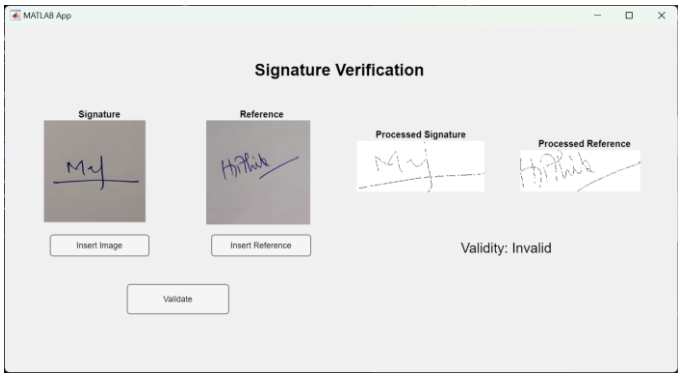
A. *Hasil*

Berikut adalah contoh dari hasil program



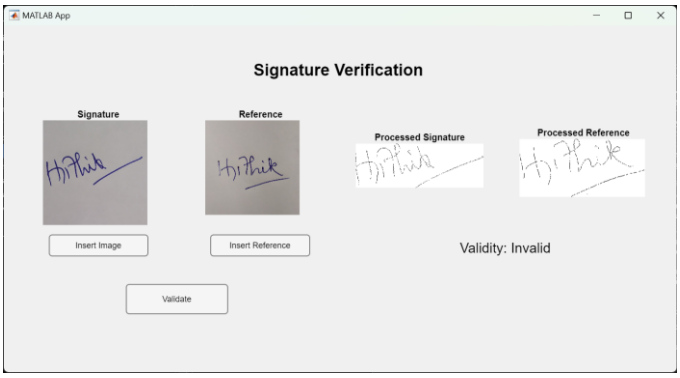
Gambar IV.1 Verifikasi valid 1

Gambar IV.2 Verifikasi valid 2

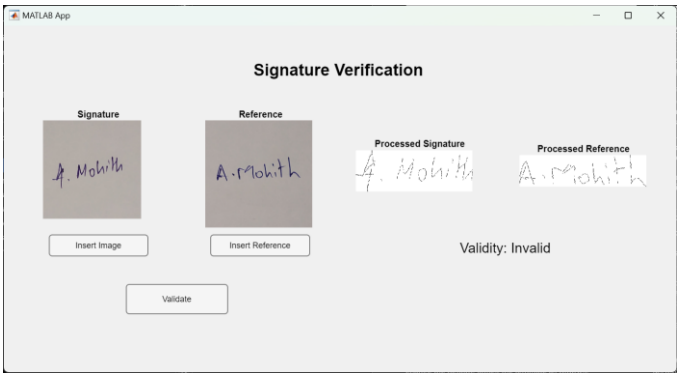


Gambar IV.3 Verifikasi invalid 1

Gambar IV.4 Verifikasi invalid 2



Gambar IV.5 Verifikasi invalid 3



Gambar IV.6 Verifikasi invalid 4

B. *Analisis*

Berdasarkan beberapa percobaan yang telah dilakukan, program dapat menentukan tanda tangan yang mirip dan berbeda dengan baik. Untuk tanda tangan yang valid, dapat dilihat pada Gambar IV.1 dan Gambar IV.2 dan untuk tanda tangan yang invalid dapat dilihat pada Gambar IV.3 dan Gambar IV.4.

Pada deteksi yang valid, kedua tanda tangan berbentuk mirip walaupun masih ada perbedaan seperti pada *zoom* dari gambar dan lengkungan yang sedikit berbeda. Untuk verifikasi invalid, keduanya berbentuk berbeda, tetapi ada kemiripan seperti pada garis bawah dan huruf ‘S’ yang serupa. Meskipun begitu, program masih dapat memvalidasi dengan baik.

Percobaan lebih lanjut dilakukan pada Gambar IV.5 dan Gambar IV.6, dimana gambar adalah tanda tangan yang sangat

mirip dari bentuknya. Program menghasilkan validasi invalid pada kedua hasil. Jika dilihat, kedua gambar memang memiliki perbedaan seperti pada *spacing* dan *font* penulisan. Hal ini menunjukkan program dapat melihat perbedaan tersebut. Namun, bisa saja perbedaan kecil tersebut hanyalah kesalahan dari pemilik tanda tangan.

V. KESIMPULAN

Program verifikasi tanda tangan yang dibuat dapat memverifikasi tanda tangan dengan baik pada data yang dicoba. Tanda tangan yang mirip diklasifikasikan sebagai valid dan yang berbeda sebagai invalid. Namun, percobaan yang dilakukan belum dapat menentukan baik atau tidaknya kinerja program karena data yang sedikit. Selain itu, *feature extraction* yang dilakukan dapat ditambah untuk mencakup kasus yang lebih banyak lagi karena pada makalah ini hanya digunakan empat fitur dan nilai *thresholding* yang digunakan masih dapat dioptimasikan lagi.

ACKNOWLEDGMENT

Terima kasih kepada Bapak Rinaldi Munir sebagai dosen pengampu Pengolahan dan Interpretasi Citra Tahun Ajaran 2021-2022 yang telah memberikan ilmu sehingga makalah ini dapat ditulis.

REFERENSI

[1] Munir, Rinaldi. 2022. Pembentukan citra dan digitalisasi citra. Bandung: https://informatika.stei.itb.ac.id/~rinaldi.munir/

[2] Munir, Rinaldi. 2022. Citra biner. Bandung: https://informatika.stei.itb.ac.id/~rinaldi.munir/

[3] Munir, Rinaldi. 2022. *Image Enhancement.* Bandung: https://informatika.stei.itb.ac.id/~rinaldi.munir/

[4] P. V. Hatkar, Z. J. Tamboli, "Image Processing for Signature Verification," IJIRCST, May 2015.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 18 Desember 2022

Muhammad Iqbal Sigid 13519152