BAB 3
ANALISIS DAN PERANCANGAN

3.1 Analisis Sistem

Dalam proses pembuatan suatu sistem mutlak dilakukan analisis terhadap sistem yang akan dibangun, analisis yang dilakukan untuk membangun aplikasi perbandingan algoritma *template matching* dan *feature extraction* pada OCR dijelaskan pada subbab-subbab berikut ini.

3.1.1 Analisis Masalah

Masalah yang timbul dalam OCR adalah tingkat akurasi pengenalan terhadap citra karakter yang akan dikenali, kompleksitas, pengembangan, dan waktu yang dibutuhkan kedua algoritma.

3.1.2 Analisis Algoritma

Tabel 3.1 adalah perbandingan algoritma *template matching* dan *feature extraction* berdasarkan proses yang dibutuhkan, kompleksitas, hasil pengenalan, pengembangan, dan waktu yang dibutuhkan masing-masing algoritma untuk mengenali dokumen uji.

Tabel 3.1. Hasil Pengukuran Kedua Algoritma.

No.	Pengukuran	*Template Matching*	*Feature Extraction*
1.	Proses yang dibutuhkan	Algoritma *template matching* membutuhkan tiga proses utama dengan penjelasan sebagai berikut: 1. Pemetaan intensitas pixel 2. Perhitungan nilai eror	Algoritma *feature extraction* membutuhkan enam proses utama dengan penjelasan sebagai berikut: 1. Perhitungan rasio citra 2. Pendeteksian keterbukaan citra 3. Perhitungan jumlah perpotongan garis tengah

No.	Pengukuran	*Template Matching*	*Feature Extraction*
		minimum 3. Pencarian nilai eror minimum	terhadap pixel hitamsecara vertikal dan horizontal. Proses ini disebut sebagai pendeteksian intersecttion vertikal dan horizontal 4. Perhitungan garis vertikal dan horizontal citra karakter. 5. Perhitungan jumlah pixel hitam di setiap blok (sembilan blok) dibagi dengan resolusi citra karakter. 6. Pengklasifikasian fitur-fitur citra yang terdeteksi
2.	Kompleksitas	Algoritma *template matching* memiliki kompleksitas 2n+1 dengan penjelasan sebagai berikut: 1. Pemetaan intensitas pixel hanya dilakukan sekali yaitu pada saat citra akan dikenali. 2. Perhitungan nilai eror minimum dilakukan pada seluruh *template*. 3. Pencarian nliai eror yang paling kecil dilakukan dengan membandingkan nilai eror masing-masing *template*. 4. Jika pemetaan intensitas pixel dihitung satu kali, perhitungan dan pencarian nilai eror minimum dihitung n-kali, maka kompleksitas algoritma template matching adalah 2n+1.	Algoritma *feature extraction* memiliki kompleksitas 2n+1 dengan penjelasan sebagai berikut: 1. Pemetaan fitur citra karakter burupa perhitungan rasio citra, pendeteksian keterbukaan citra, perhitungan perpotongan pixel hitam terhadap garis tengah secara vertikal dan horizontal, perhitungan garis vertikal dan horizontal, dan perhitungan pixel hitam pada masing-masing blok citra hanya dilakukan saat citra karakter akan dikenali. 2. Pengklasifikasian terbagi menjadi dua tahap yaitu perhitungan jarak dan pengurutan jarak secara *ascending*. Kedua tahap tersebut dilakukan sebanyak citra template yang disimpan pada basis data. 3. Jika proses pemetaan fitur-fitur citra dilakukan hanya saat citra karakter akan dikenali, perhitungan jarak dan pengurutan dilakukan sebanyak citra *template*, maka kompleksitas algoritma feature extraction adalah 2n+1
3.	Pengembangan	Algoritma *template matching* sulit dikembangkan, karena algoritma template matching bekerja dengan	Algoritma *feature extraction* memiliki peluang untuk bisa dikembangkan terutama pada ciri-ciri khusus citra karakter. Salah satu contohnya adalah fitur stroke

No.	Pengukuran	*Template Matching*	*Feature Extraction*
		menyesuaikan intensitas pixel.	(jumlah garis yang membentuk karakter)
4.	Hasil pengenalan	Berdasarkan gambar 4.25, algoritma *template matching* memiliki pengenalan lebih buruk dibandingkan dengan algoritma *feature extraction*.	Berdasarkan gambar 4.25, algoritma *feature extraction* memiliki pengenalan lebih baik dibandingkan dengan algoritma *template matching*.
5.	Waktu	Berdasarkan gambar 4.24, algoritma *template matching* membutuhkan waktu yang lebih lama dibandingkan dengan algoritma *feature extraction* untuk mengenali citra karakter.	Berdasarkan gambar 4.24, algoritma *feature extraction* membutuhkan waktu yang lebih singkat dibandingkan dengan algoritma *template matching* untuk mengenali citra karakter.

Berdasarkan proses yang dibutuhkan kedua algoritma, algoritma *feature extraction* membutuhkan proses yang lebih banyak dibandingkan dengan algoritma *template matching*, yaitu proses pendeteksian keterbukaan citra, pendeteksian perpotongan garis tengah citra, perhitungan jumlah pixel hitam tiap blok, perhitungan jumlah garis vertikal dan horizontal, dan proses klasifikasi. Sedangkan algoritma *template matching* hanya membutuhkan proses pemetaan intensitas pixel citra karakter yang akan dikenali, perhitungan eror, dan pencarian nilai eror minimum.

Kompleksitas algoritma dihitung berdasarkan operasi dasar yang dilakukan kedua algoritma. Operasi dasar yang dilakukan pada algoritma *template matching* adalah sebagai berikut:

1. Pemetaan intensitas pixel. Pemetaan ini dilakukan satu kali saat citra karakter akan dikenali.
2. Perhitungan nliai eror minimum. Perhitungan yang dilakukan sebanyak citra template dinotasikan dengan n.

3. Pencarian nilai eror minimum dilakukan pada seluruh citra template. Jumlah pencarian yang dilakukan terhadap seluruh citra template dinotasikan dengan n.
4. Tabel 3.1 adalah perhitungan jumlah operasi dasar yang dilakukan pada algoritma *template matching*.

Tabel 3.2. Perhitungan Jumlah Operasi Dasar Algoritma *Template Matching*.

No.	Operasi dasar	Jumlah iterasi
1.	Pemetaan pixel	1 kali
2.	Perhitungan nilai eror minimum	n kali
3.	Pencarian nliai eror terkecil	n kali
Total		**2n+1**

5. Berdasarkan hasil perhitungan jumlah operasi dasar algoritma template matching pada tabel 3.2, maka kompleksitas algoritma template matching adalah 2n+1.

Operasi dasar yang dilakukan pada algoritma *feature extraction* adalah sebagai berikut:

1. Pemetaan fitur citra karakter. Pemetaan ini dilakukan satu kali saat citra karakter akan dikenali.
2. Perhitungan jarak dengan menggunakan persamaan 2-7. Perhitungan yang dilakukan sebanyak citra template dinotasikan dengan n.
3. Pencarian jarak minimum dilakukan pada seluruh citra template. Jumlah pencarian yang dilakukan terhadap seluruh citra template dinotasikan dengan n.
4. Tabel 3.3 adalah perhitungan jumlah operasi dasar yang dilakukan pada algoritma *feature extraction*.

Tabel 3.3. Perhitungan Jumlah Operasi Dasar Algoritma *Feature Extraction*.

No.	Operasi dasar	Jumlah iterasi
1.	Pemetaan fitur citra karakter	1 kali
2.	Perhitungan jarak	n kali
3.	Pencarian jarak terkecil	n kali
Total		**2n+1**

5. Berdasarkan hasil perhitungan jumlah operasi dasar *feature extraction* pada tabel 3.3, maka kompleksitas algoritma *feature extraction* adalah 2n+1.

Berdasarkan hasil perhitungan kompleksitas kedua algoritma, algoritma *feature extraction* dan algoritma *template matching* memiliki kompleksitas yang sama yaitu 2n+1.

Dari lima proses pemetaan fitur pada algoritma *feature extraction*, masih ada kemungkinan penambahan fitur-fitur khusus citra karakter. Salah satu contohnya adalah fitur jumlah *stroke* (jumlah garis yang membentuk karakter). Sedangkan pada algoritma *template matching*, kecil kemungkinan penambahan fitur karena algoritma *template matching* bekerja berdasarkan pemetaan intensitas pixel.

Berdasarkan hasil pengenalan, algoritma *feature extraction* dapat mengenali citra karakter lebih baik dibandingkan dengan algoritma *template matching*. Hal ini terlihat pada gambar 4.25 yang mengilustrasikan persentase hasil pengenalan kedua algoritma terhadap dokumen uji.

Algoritma *feature extraction* membutuhkan waktu yang lebih singkat dibandingkan dengan algoritma *template matching* dalam mengenali citra karakter yang terlihat pada gambar 4.24. Hal ini terjadi karena pada algoritma *template matching*, setiap *template* karakter memiliki 120 deretan pixel yang harus dibandingakan dengan 120 deretan pixel citra yang akan dikenali. Sedangkan pada algoritma *feature extraction*, terdapat 18 fitur yang harus dibandingkan.

Berdasarkan tabel 3.1, algoritma *feature extraction* lebih unggul dibandingkan dengan algoritma *template matching*. Keunggulan tersebut terlihat

pada hasil pengukuran kedua algoritma berdasarkan hasil pengenalan, pengembangan, dan waktu.

3.1.3 Analisis Proses

Analisis proses ini akan menjelaskan mengenai proses yang digunakan dalam aplikasi perbandingan algoritma *template matching* dan *feature extraction* pada OCR.

3.1.3.1 Proses *Preprocessing*

Proses preprocessing terdiri dari proses *grayscaling* dan binerisasi.

3.1.3.1.1 Proses *Grayscaling*

Untuk mengubah citra berwarna yang memiliki nilai matrik masing-masing R, G, dan B menjadi citra grayscale dengan membagi jumlah ketiga *layer* seperti pada persamaan 3-1.

$$Ko = \frac{Ri}{} \quad \text{.....(3-1)}$$

Ko adalah nilai *layer* baru yang didapatkan dari hasil perhitungan ketiga *layer* citra. Contoh perhitungan proses *grayscaling* dijelaskan sebagai berikut.

1. Terdapat satu pixel yang memiliki intensitas yang berbeda pada setiap layernya.

   R = 120; G = 50 ; B = 270;

2. Penjumlahan dari ketiga layer (R, G, dan B) menggunakan persamaan 3-1. Pembagian dilakukan dengan menggunakan operator *div* karena intensitas pixel bertipe bilangan bulat.

Ko = (120 + 50 + 270) div 3

3. Berikut ini adalah hasil perhitungan berdasarkan persamaan 3-1.

Ko = 146

4. Nilai 146 adalah intensitas baru yang dihasilkan dari proses *grayscaling*.

Hasil dari proses grayscale adalah citra dengan intensitas *pixel* sebanyak 256 intensitas. Nilai intensitas paling rendah merepresentasikan warna hitam dan intensitas paling tinggi merepresentasikan warna putih.

3.1.3.1.2 Proses Binerisasi

Proses ini dilakukan untuk mengubah citra berderajat keabuan menjadi citra biner atau hitam putih. Intensitas *pixel* citra hasil *grayscale* dibandingkan dengan nilai ambang (*thresshold*). Jika nilai *pixel* lebih besar dari nilai ambang, maka *pixel* akan direpresentasikan dengan warna putih. Jika nilai *pixel* lebih kecil daripada nilai ambang, maka *pixel* akan direpresentasikan dengan warna hitam.

3.1.3.2 Proses Pensegmentasian

Proses pensegmentasian terdiri dari segmentasi baris dan segmentasi karakter. Segmentasi karakter dilakukan dengan menggunakan metode *Connected component analysis*.

3.1.3.2.1 Proses Segmentasi Baris

Segmentasi baris dilakukan dengan cara menjumlahkan angka 0 atau 1 yang mewakili nilai *pixel* suatu citra. Penjumlahan ini dilakukan pada setiap baris *pixel* citra. Jika hasil penjumlahan suatu baris *pixel* bernilai 0, maka pada baris

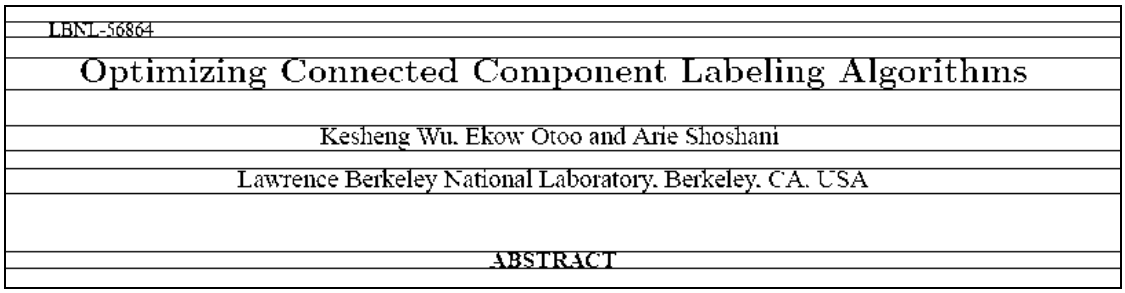
pixel tersebut tidak ada karakter yang terdeteksi. Jika hasil penjumlahan suatu baris *pixel* lebih dari 0, maka pada baris *pixel* tersebut terdapat karakter.

Gambar 3.1 mengilustrasikan perhitungan pixel per baris yang digunakan untuk segmentasi baris.

baris *pixel*											jumlah *pixel*
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	0	0	0	0	3
2	0	0	0	1	1	1	0	0	0	0	3
3	0	0	0	1	1	1	1	0	0	0	4
4	0	0	1	1	0	1	1	0	0	0	4
5	0	0	1	1	0	0	1	0	0	0	3
6	0	0	1	0	0	0	1	1	0	0	3
7	0	1	1	1	0	0	1	1	0	0	5
8	0	1	1	1	1	1	1	1	0	0	7
9	1	1	1	0	0	0	0	1	1	0	5
10	1	1	0	0	0	0	0	0	1	1	4
11	1	1	0	0	0	0	0	0	1	1	4
12	1	0	0	0	0	0	0	0	1	1	3
13	0	0	0	0	0	0	0	0	0	0	0

Gambar 3.1. Perhitungan Baris Nilai *Pixel*.

Garis hitam melintang dari kiri ke kanan pada gambar 3.2 yang terletak di atas dan di bawah setiap karakter merupakan garis yang digunakan untuk mengilustrasikan segmentasi baris. Jika tinggi baris kurang dari enam pixel, maka akan dianggap sebagai noise.


LBNL-56864

Optimizing Connected Component Labeling Algorithms

Kesheng Wu. Ekow Otoo and Arie Shoshani

Lawrence Berkeley National Laboratory. Berkeley. CA. USA

ABSTRACT

Gambar 3.2. Citra Hasil Segmentasi Baris.

3.1.3.2.2 Proses Segmentasi Karakter

Segmentasi karakter dilakukan dengan menggunakan metode *Connected component analysis*. *Connected component analysis* atau CCA bekerja dengan cara memeriksa tetangga dari pixel yang sedang dianalisis. Pada penelitian ini, teknik analisis yang digunakan adalah delapan tetangga.

3.1.3.3 Proses Penormalisasian

Proses normalisasi terdiri proses pencarian batas yang digunakan untuk proses penskalaan.

3.1.3.3.1 Proses Pencarian Batas

Setelah karakter disegmentasi, maka karakter akan dicari batas atas, bawah, kiri, dan kanannya. Pencarian batas digunakan untuk mengetahui resolusi karakter. Batas-batas yang ada akan digunakan pada saat proses penskalan pengambilan ciri-ciri karakter.

3.1.3.3.2 Proses Penskalaan

Penskalaan dilakukan jika karakter hasil segmentasi memiliki tinggi lebih atau kurang dari 12 *pixel* dan lebar lebih atau kurang dari 10 *pixel* dan mutlak dilakukan untuk proses pengenalan dengan metode *template matching*. Hal ini dilakukan karena citra *template* memiliki resolusi 10x12 *pixel*.

Langkah-langkah berikut ini digunakan untuk proses penskalaan.

1. Gambar 3.3 adalah pixel yang dilengkapi dengan koordinatnya sebelum pixel tersebut mengalami proses penskalaan.

(0,0)	(1,0)	(2,0)	(3,0)	(4,0)

Gambar 3.3. Pixel Awal Sebelum Proses Penskalaan.

2. Gambar 3.4 adalah ilustrasi pixel target hasil proses penskalaan.

(0,0)	(1,0)	(2,0)

Gambar 3.4. Pixel Hasil Proses Penskalaan.

3. Tabel 3.4 adalah perhitungan proses penskalaan dengan menggunakan persamaan 2-4 dan persamaan 2-5.

Tabel 3.4. Perhitungan Proses Penskalaan.

Koordinat Pixel Awal		**Faktor Pengali**		**Koordinat Pixel Target**	
I	**j**	**dx**	**dy**	**x=round(i*dx)**	**y=round(j*dy)**
0	0	0.5	0.5	0	0
1	0	0.5	0.5	1	0
2	0	0.5	0.5	1	0
3	0	0.5	0.5	2	0
4	0	0.5	0.5	2	0

4. Koordinat pixel target x dan y adalah koordinat hasil pemetaan dari pixel lama terhadap pixel target.

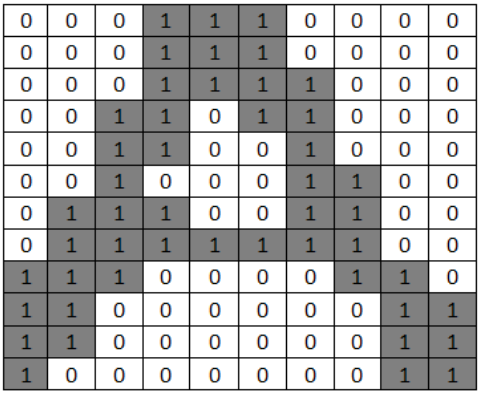
3.1.3.4 Proses Pengenalan

Proses pengenalan dilakukan oleh algoritma *template matching* dan *feature extraction*. Pembahasan mengenai kedua algoritma ini terdapat pada sub bab analisis algoritma.

3.1.3.4.1 Pengenalan dengan Algoritma *Template Matching*

Pengenalan pola dengan menggunakan metode *template matching* dilakukan dengan cara membandingkan citra masukan dengan citra *template*. Citra masukan dihitung berdasarkan banyaknya titik yang sesuai dengan citra *template*.

Pixel citra biner ditelusuri mulai dari kiri atas hingga ke kanan bawah. Citra biner dengan *pixel* berwarna hitam akan direpresentasikan dengan nilai 1. Sedangkan *pixel* citra yang berwarna putih akan direpresentasikan dengan nilai 0. Gambar 3.5 adalah gambar yang mengilustrasikan angka 1 dan 0 yang mewakili nilai *pixel* citra.



Gambar 3.5. Citra Hitam Putih dengan Nilai *Pixel* 1 dan 0.

Deretan angka biner pada citra masukan akan dihitung dengan deretan angka biner pada citra *template*. *Template* dengan nilai eror terkecil merupakan *template* citra yang paling sesuai dengan citra masukan.

Contoh berikut ini adalah penerapan pengenalan karakter dengan menggunakan metode *template matching* yang menggunakan persamaan 2-6.

1. Tabel 3.5 adalah tabel contoh deretan angka biner citra karakter *template*.

Tabel 3.5. Karakter dan Deretan Angka Pola Pixel.

Karakter	**Deretan angka biner citra *template***
A	0 0 1 1 1 0 0 0 0
B	1 1 1 0 0 0 1 1 0
C	1 0 1 0 0 0 1 1 1
D	0 1 1 1 0 1 0 0 1

2. Deretan berikut ini adalah angka biner citra karakter masukan.

   0 0 0 1 0 1 0 1 0

3. Tabel 3.6 adalah perhitungan eror dengan menggunakan persamaan 2-6:

Tabel 3.6. Perhitungan Nilai Eror.

Karakter	Perhitungan nilai eror
A	(0-0)^2 + (0-0)^2 + (0-1)^2 + (1-1)^2 + (0-1)^2 + (1-0)^2 + (0-0)^2 + (1-0)^2 + (0-0)^2 = 4
B	(0-1)^2 + (0-1)^2 + (0-1)^2 + (1-0)^2 + (0-0)^2 + (1-0)^2 + (0-1)^2 + (1-1)^2 + (0-0)^2 = 6
C	(0-1)^2 + (0-0)^2 + (0-1)^2 + (1-0)^2 + (0-0)^2 + (1-0)^2 + (0-1)^2 + (1-1)^2 + (0-1)^2 = 6
D	(0-0)^2 + (0-1)^2 + (0-1)^2 + (1-1)^2 + (0-0)^2 + (1-0)^2 + (0-0)^2 + (1-0)^2 + (0-1)^2 = 5

Jika nilai eror masing-masing *template* sudah diketahui, maka cari nilai eror terkecil. Nilai eror terkecil adalah *template* yang paling sesuai dengan karakter. Nilai terkecil dari hasil perhitungan pada tabel 3.6 adalah 4. Maka citra *template* yang paling sesuai dengan citra masukan adalah citra karakter A.

3.1.3.4.2 Pengenalan dengan Algoritma *Feature Extraction*

Algoritma *feature extraction* merupakan salah satu cara untuk mengenali suatu objek dengan melihat ciri-ciri khusus yang dimiliki objek tersebut. Tujuan dari *feature extraction* adalah melakukan perhitungan dan perbandingan yang bisa digunakan untuk mengklasifikasikan ciri-ciri yang dimiliki oleh suatu citra.

Sebelum mengambil ciri dari suatu citra karakter, maka citra karakter tersebut harus dicari batas kanan, kiri, atas, dan bawahnya terlebih dahulu. Ciri-ciri yang diambil dari citra karakter *template* maupun citra karakter masukan adalah perbandingan lebar dan tinggi citra karakter, jumlah garis horizontal dan garis vertikal, perpotongan *pixel* hitam terhadap garis vertikal dan garis horizontal

di tengah citra. Citra karakter dianalisis apakah terbuka ke kanan, kiri, atas, atau bawah. Karakter citra akan dibagi menjadi sembilan bidang simetris kemudian *pixel* yang berwarna hitam pada masing-masing bidang akan dibagi dengan jumlah *pixel* keseluruhan.

3.1.3.4.2.1 Perbandingan Lebar dan Tinggi

Perbandingan/rasio lebar dan tinggi citra karakter dihitung dengan cara membagi lebar dan tinggi citra. Lebar citra didapatkan dari selisih antara batas kanan dan kiri citra. Sedangkan tinggi citra didapatkan dari selisih antara batas bawah dan atas citra.

$$\text{ratio} = \frac{\text{batas kanan} - b}{\text{batas bawah} - b} \quad \text{.....(3-2)}$$

Jika suatu citra hanya memiliki batas kiri sama dengan batas kanan dan batas atas sama dengan batas bawah, maka batas kanan adalah batas kiri yang ditambah 1 *pixel* sedangkan batas bawah adalah batas atas ditambah dengan 1 *pixel*.

3.1.3.4.2.2 Jumlah Garis Horizontal

Gambar 3. 6 adalah citra karakter yang memiliki garis horizontal.



Gambar 3.6. Citra Karakter dengan Garis Horizontal dan Garis Vertikal.

Setiap karakter diperiksa apakah memiliki ciri berupa garis horizontal dengan cara berikut ini:

1. Telusuri citra mulai dari kiri atas ke kanan bawah.
2. Jika ditemukan *pixel* berwarna hitam, maka *pixel* tersebut akan diwakili dengan angka 1. Sedangkan jika ditemukan *pixel* berwarna putih, maka *pixel* tersebut akan diwakili dengan angka 0.
3. Di akhir baris setiap *pixel*, jumlahkan angka yang mewakili *pixel*. Jumlah pixel akan menjadi 0 jika ditemukan pixel putih sebelum jumlah pixel mencapai 80% lebar citra karakter.
4. Jika posisi *scan* sudah mencapai kanan bawah, maka periksa jumlah *pixel* di masing-masing baris.
5. Jika jumlah *pixel* lebih dari atau sama dengan 80% lebar citra, maka citra karakter tersebut memiliki ciri berupa garis horizontal.
6. Hitung jumlah garis horizontal yang ada pada citra karakter.

3.1.3.4.2.3 Jumlah Garis Vertikal

Gambar 3.7 adalah citra karakter yang memiliki ciri berupa garis vertikal.



Gambar 3.7. Citra Karakter dengan Garis Vertikal.

Langkah-langkah berikut ini digunakan untuk mencari jumlah garis vertikal suatu citra karakter.

1. *Scan* citra mulai dari kiri atas ke kanan bawah.
2. Jika ditemukan *pixel* berwarna hitam, maka *pixel* tersebut akan diwakili dengan angka 1. Sedangkan jika ditemukan *pixel* berwarna putih, maka *pixel* tersebut akan diwakili dengan angka 0.

3. Di akhir kolom setiap *pixel*, jumlahkan angka yang mewakili *pixel*. Jumlah pixel akan menjadi 0 jika ditemukan pixel putih sebelum jumlah pixel mencapai 80% lebar citra karakter.
4. Jika posisi *scan* sudah mencapai kanan bawah, maka periksa jumlah *pixel* di masing-masing kolom.
5. Jika jumlah *pixel* lebih dari atau sama dengan 80% tinggi citra, maka citra karakter tersebut memiliki ciri berupa garis vertikal.
6. Hitung jumlah garis vertikal yang ada pada citra karakter.

3.1.3.4.2.4 Ciri Citra Karakter Terbuka

Suatu citra karakter diperiksa apakah terbuka ke kiri, kanan, atas, atau bawah. Gambar 3.8 adalah citra karakter yang memiliki ciri-ciri berupa terbuka ke kiri, kanan, atas, atau bawah.



Gambar 3.8. Citra Karakter dengan Ciri-Ciri Terbuka.

Langkah-langkah berikut ini digunakan untuk mendeteksi apakah suatau citra karakter terbuka ke atas atau ke bawah.

1. *Scan* citra dari atas hingga mencapai 40% tinggi citra. Nilai 40% didapatkan dari hasil penelitian.
2. Jika ditemukan *pixel* berwarna hitam, maka *pixel* tersebut akan diwakili dengan angka 1. Sedangkan jika ditemukan *pixel* berwarna putih, maka *pixel* tersebut akan diwakili dengan angka 0.
3. Jumlahkan angka *pixel* pada masing-masing kolom dari kiri ke kanan.

4. Periksa jumlah pada masing-masing kolom. Jika ditemukan pola jumlah *pixel* pada suatau kolom sama dengan 0 dan jumlah *pixel* pada kolom di sebelahnya lebih dari 0 sebanyak dua kali, maka citra karakter tersebut dinyatakan terbuka ke atas.
5. Hal yang sama dilakukan untuk mendeteksi citra karakter terbuka ke bawah, namun citra di-*scan* mulai dari 70% tinggi citra hingga mencapai 100% tinggi citra. Nilai 70% didapatkan dari hasil penelitian.

Langkah-langkah berikut ini digunakan untuk mendeteksi apakah suatu citra karakter terbuka ke kiri atau ke kanan.

1. *Scan* citra dari kiri hingga mencapai 30% lebar citra. Nilai 30% didapatkan dari hasil penelitian.
2. Jika ditemukan *pixel* berwarna hitam, maka *pixel* tersebut akan diwakili dengan angka 1. Sedangkan jika ditemukan *pixel* berwarna putih, maka *pixel* tersebut akan diwakili dengan angka 0.
3. Jumlahkan angka *pixel* pada masing-masing baris dari atas ke bawah.
4. Periksa jumlah *pixel* pada masing-masing baris. Jika ditemukan pola jumlah *pixel* pada suatu baris sama dengan 0 dan jumlah *pixel* pada baris di bawahnya lebih dari 0 sebanyak dua kali, maka citra karakter tersebut terbuka ke kiri.
5. Hal yang sama dilakukan untuk mendeteksi citra karakter terbuka ke kanan, namun citra di-*scan* mulai dari 70% lebar citra hingga mencapai 100% lebar citra. Nilai 70% didapatkan dari hasil penelitian

3.1.3.4.2.5 Jumlah Perpotongan Garis Vertikal Di Tengah Citra

Pada citra karakter ditarik suatu garis vertikal yang tepat memotong bagian tengah citra. Garis vertikal tersebut ditelusuri dan apabila menemukan *pixel* berwarna hitam dan *pixel* di bawahnya berwarna putih, maka dihitung satu perpotongan. Gambar 3.9 adalah citra karakter dengan tiga perpotongan terhadap garis vertikal.



Gambar 3.9. Perpotongan Garis Vertikal di Bagian Tengah Citra.

3.1.3.4.2.6 Jumlah Perpotongan Garis Horizontal Di Tengah Citra

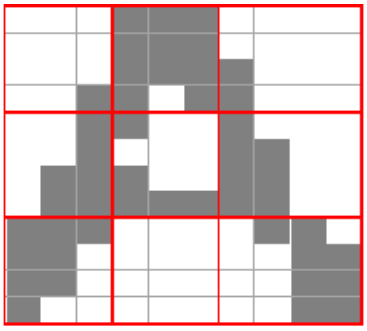
Pada citra karakter ditarik suatu garis horizontal yang tepat memotong bagian tengah citra. Garis horizontal tersebut ditelusuri dan apabila menemukan *pixel* berwarna hitam dan *pixel* di sebelahnya berwarna putih, maka dihitung satu perpotongan. Gambar 3.10 adalah citra karakter dengan dua perpotongan terhadap garis horizontal.



Gambar 3.10. Perpotongan Garis Horizontal di Bagian Tengah Citra.

3.1.3.4.2.7 Perbandingan *Pixel* Hitam Di Setiap Blok Citra

Citra karakter dibagi menjadi sembilan bagian simetris dan dihitung perbandingan jumlah *pixel* hitam pada masing-masing bagian tersebut dengan jumlah seluruh *pixel*. Gambar 3.11 adalah ilustrasi pembagian blok citra karakter.



Gambar 3.11. Citra karakter yang dibagi menjadi sembilan blok.

Bagian yang pertama dihitung adalah blok pada bagian kiri atas citra karakter. Jika ditemukan *pixel* berwarna hitam, maka *pixel* tersebut akan diwakili dengan angka 1. Sedangkan jika ditemukan *pixel* berwarna putih, maka *pixel* tersebut akan diwakili dengan angka 0. Jumlahkan angka-angka *pixel* yang terdapat di blok pertama dan dibagi dengan resolusi citra karakter.

Hal yang sama dilakukan pada seluruh blok. Blok kedua terletak di bawah blok pertama. Blok ketiga terletak di bawah bagian blok kedua. Blok keempat terletak di sebelah kanan blok pertama dan begitu seterusnya hingga mencapai blok terakhir yaitu blok kesembilan yang berada di bagian kanan bawah citra.

3.1.3.4.2.8 Klasifikasi

Jika seluruh ciri-ciri citra karakter sudah didapatkan, maka ciri-ciri tersebut diklasifikasikan dengan ciri-ciri yang ada pada basis data. Metode klasifikasi yang digunakan pada penelitian ini adalah metode k-NN atau k *Nearest Neighbour*.

3.1.3.4.2.9 k-Nearest Neighbour

Algoritma k-*Nearest Neighbour* mengenali citra berdasarkan ciri-ciri yang paling mendekati antara citra masukan dengan citra *template*. Tujuannya adalah mengklasifikasikan citra karakter berdasarkan atribut yang dimilikinya.

Berikut ini adalah contoh perhitungan k-NN untuk mengenali karakter.

1. Tabel 3.7 adalah tabel citra karakter *template* yang dilengkapi dengan ciri-cirinya.

Tabel 3.7. Citra Karakter *Template* dan Atributnya.

no	kar	ratio	t_kiri	t_kanan	t_atas	t_bawah	int_v	int_h	garis_v	garis_h	Blok1
1	A	0.727273	0	0	0	1	2	2	0	1	0
2	B	0.727273	0	1	0	0	3	1	1	3	0.068182
3	C	0.818182	0	1	0	0	2	1	0	0	0.030303
4	D	0.818182	0	0	0	0	2	2	1	0	0.060606

2. Tabel 3.8 adalah ciri-ciri citra karakter masukan yang akan dikenali.

Tabel 3.8. Citra Karakter Masukan dan Atributnya.

no	kar	ratio	t_kiri	t_kanan	t_atas	t_bawah	int_v	int_h	garis_v	garis_h	Blok1
1	?	0.818182	0	0	0	1	2	2	0	1	0.010101

3. Tentukan faktor k. Faktor k menyatakan jumlah objek baru hasil pengenalan yang akan diambil. Jika k bernilai 2, maka dua urutan teratas akan diambil. Pada pengenalan karakter ini, k yang digunakan bernilai 1, maka hanya nilai hasil perhitungan terkecil yang akan diambil.
4. Tabel 3.9 adalah perhitungan k-NN yang dilakukan dengan persamaan 2-7 dengan menggunakan atribut citra *template* dan citra masukan.

Tabel 3.9. Perhitungan Jarak untuk Klasifikasi.

No	Kar	Perhitungan	Hasil Perhitungan
1	A	(0.727273-0.818182)^2 + (0-0)^2 + (0-0)^2 + (0-0)^2 + (1-1)^2 + (2-2)^2 + (2-2)^2 + (0-0)^2 + (1-1)^2 + (0-0.010101)^2	0.000102
2	B	(0.727273-0.818182)^2 + (0-0)^2 + (1-0)^2 + (0-0)^2 + (0-1)^2 + (3-2)^2 + (1-2)^2 + (1-0)^2 + (3-1)^2 + (0-0.068182)^2	0.003373
3	C	(0.727273-0.818182)^2 + (0-0)^2 + (1-0)^2 + (0-0)^2 + (0-1)^2 + (2-2)^2 + (1-2)^2 + (0-0)^2 + (0-1)^2 + (0-0.030303)^2	0.000408
4	D	(0.727273-0.818182)^2 + (0-0)^2 + (0-0)^2 + (0-0)^2 + (0-1)^2 + (2-2)^2 + (2-2)^2 + (1-0)^2 + (0-1)^2 + (0-0.060606)^2	0.002551

5. Urutkan hasil perhitungan dari kecil ke besar. Tabel 3.7 adalah tabel yang berisi hasil perhitungan dengan menggunakan persamaan 2-7 dan diurutkan secara *ascending*.

Tabel 3.10. Hasil Perhitungan Jarak.

No	Kar	Hasil Perhitungan
1	A	0.000102
3	C	0.000408
4	D	0.002551
2	B	0.003373

6. Berdasarkan parameter k, maka hanya ada satu data yang diambil yaitu karakter A.

3.1.4 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan nonfungsional adalah suatu analisis untuk mengetahui elemen-elemen yang berhubungan dengan sistem yang sedang berjalan.

3.1.4.1 Analisis Kebutuhan Perangkat Keras

Adapun spesifikasi perangkat keras minimum yang dibutuhkan untuk membangun aplikasi ini adalah sebagai berikut:

1. Prosessor AMD 1.6 Ghz.
2. Media penyimpanan (*Harddisk*) 20 GB.
3. Memori *RAM* 256 Mb.
4. Kartu grafis 32 Mb.
5. Monitor 15 inchi.

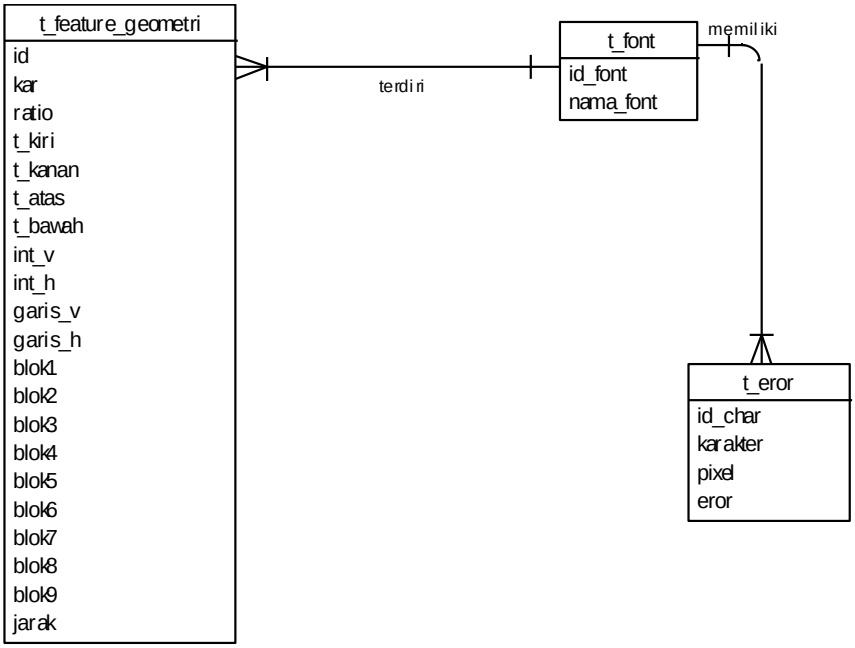
3.1.4.2 Analisis Perangkat Lunak

Berikut ini adalah spesifikasi perangkat lunak minimum yang dibutuhkan dalam membangun aplikasi:

1. Sistem operasi Window XP SP2.
2. Borland Delphi 7.
3. Wamp Server 2.0.
4. SQLyog Enterprise 8.05.
5. ODBC 5.1.

3.1.4.3 Analisis Basis Data

Basis data digunakan untuk menyimpan deretan *pixel* dan ciri-ciri karakter *template*. Pada pembangunan aplikasi ini, basis data juga digunakan untuk menghitung nilai eror pada metode *template matching* dan menghitung jarak pada metode *feature extraction*. Gambar 3.12 adalah *Entity Relation Diagram* (ERD) yang digunakan pada pembangunan aplikasi.



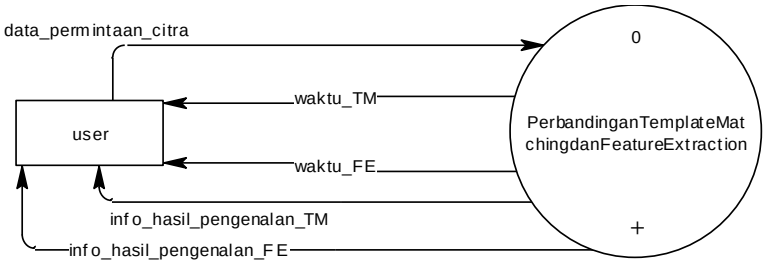
Gambar 3.12. ERD yang Digunakan pada Aplikasi Perbandingan Algoritma *Template Matching* dan Feature Extractin pada OCR.

3.1.5 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional ini meliputi diagram konteks dan *Data Flow Diagram* (DFD).

3.1.5.1 Diagram Konteks Perbandingan Algoritma *Feature extraction* dan *Template matching*

Diagram konteks adalah diagram yang menunjukkan keterhubungan antara perangkat lunak dengan konteks eksternal di luar program. Gambar 3.13 adalah diagram konteks yang dibangun dalam sistem.

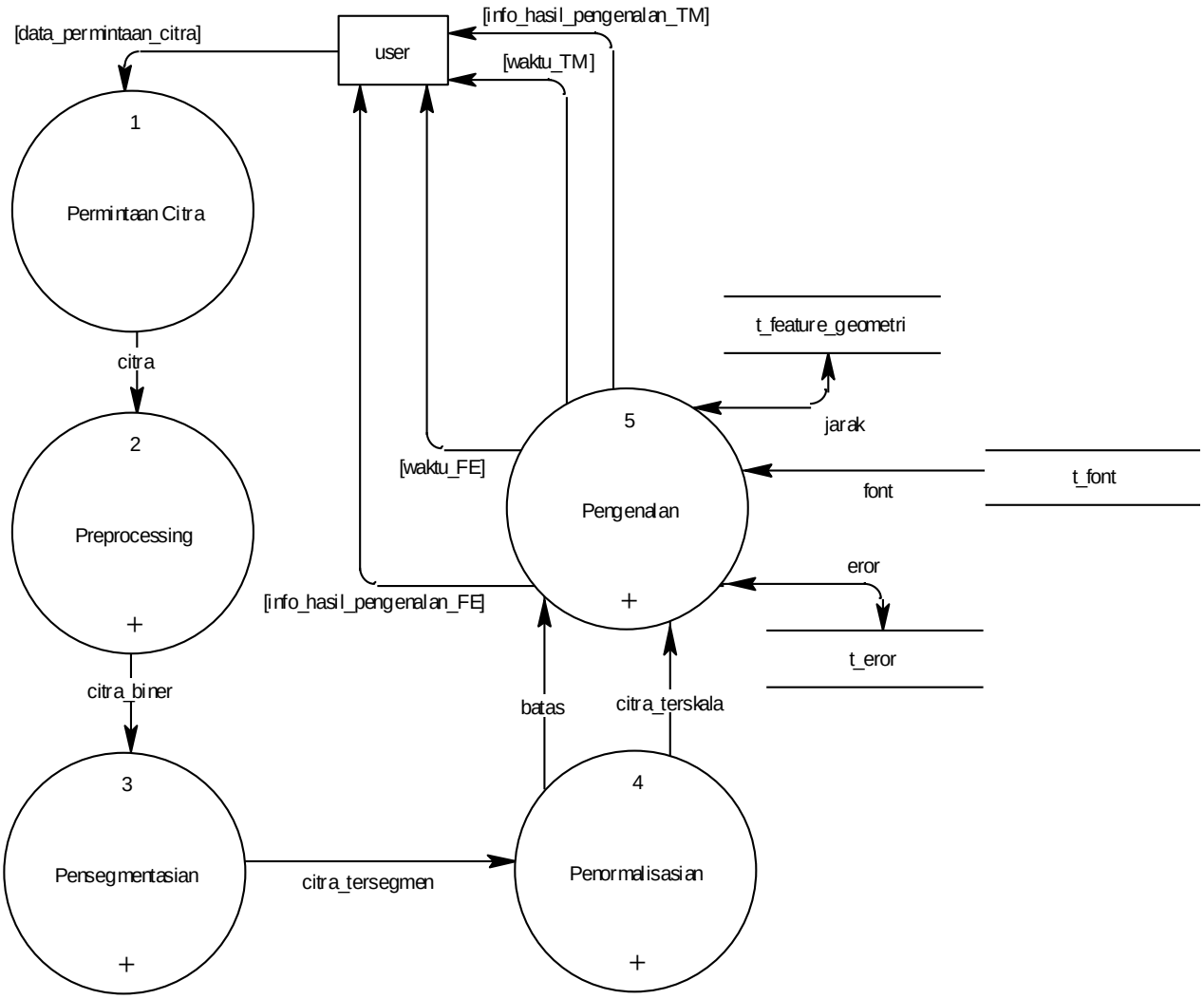


Gambar 3.13. Diagram Konteks Perbandingan Algoritma *Feature Extraction* dan *Template Matching*.

Pada gambar 3.13, waktu_TM adalah waktu yang dibutuhkan algoritma *template matching* untuk mengenali dokumen uji. waktu_FE adalah waktu yang dibutuhkan algoritma *feature extraction* untuk mengenali dokumen uji. info_hasil_pengenalan_TM adalah hasil pengenalan algoritma *template matching* terhadap dokumen uji. info_hasil_pengenalan_FE adalah hasil pengenalan algoritma *feature extraction* terhadap dokumen uji.

3.1.5.2 Data Flow Diagram Level 1

Dari diagram konteks sebelumnya, maka selanjutnya proses diuraikan menjadi bagian-bagian yang lebih kecil. Dalam aplikasi perbandingan metode ini terdapat empat proses utama yaitu *preprocessing*, segmentasi, normalisasi, dan pengenalan. Proses tersebut dapat diuraikan dengan DFD level 1 yang terdapat pada gambar 3.14.



Gambar 3.14. DFD level 1 Perbandingan Algoritma *Feature extraction* dan *Template matching*.

Tabel 3.11 adalah tabel spesifikasi proses pada DFD level 1 yang terdapat pada gambar 3.14.

Tabel 3.11. Spesifikasi Proses DFD level 1.

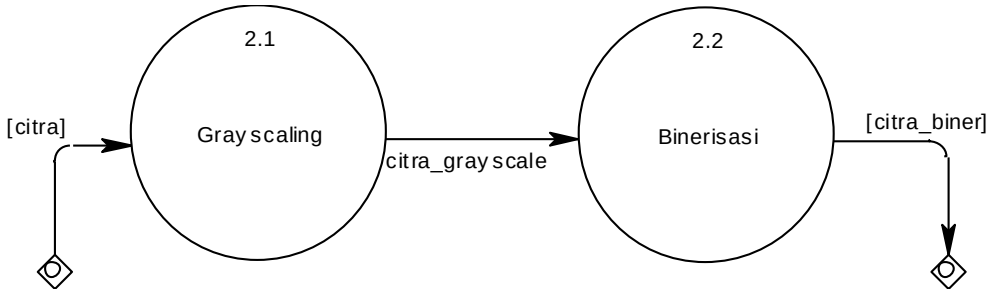
No.	Proses	Keterangan
1.	No. Proses	1
	Nama Proses	Permintaan Citra
	Input	data_permintaan_citra
	Output	Citra
	Logika proses	1. Jika user memilih menu Open, maka akan muncul dialog untuk memilih *file* citra yang akan dikenali. 2. File citra yang bisa dimuat hanya citra dengan format .bmp dan .jpg.

No.	Proses	Keterangan
	Keterangan proses	Proses ini digunakan untuk memuat citra
2.	No. Proses	2
	Nama Proses	*Preprocessing*
	Input	citra
	Output	citra_biner
	Logika proses	1. Citra dengan tiga layer diubah menjadi satu layer dengan menggunakan persamaan 3-1. 2. Pixel citra dibinerisasi dengan menggunakan nilai ambang. Jika intensitas pixel lebih besar dari nilai ambang, maka pixel akan berwarna putih dan sebaliknya.
	Keterangan proses	Pemrosesan citra dengan mengubah intensitas pixel
3.	No. Proses	3
	Nama Proses	Pensegmentasian
	Input	citra_biner
	Output	citra_tersegmen
	Logika proses	1. Jika user memilih menu pensegmentasian, maka citra karakter akan disegmentasi. Tahap pertama adalah segmentasi baris kemudian segmentasi karakter. 2. Segmentasi baris dilakukan dengan cara menjumlah intensitas pixel di setiap baris citra. Jika hasil penjumlahan lebih besar dari 0, maka ada objek yang terdeteksi. 3. Segmentasi karakter yang dilakukan dengan *connected component analysis*.
	Keterangan proses	Proses segmentasi dilakukan untuk memisahkan masing-masing citra karakter yang akan dikenali dari latar belakang dan citra karakter lain.
4.	No. Proses	4
	Nama Proses	Penormalisasian
	Input	citra_tersegmen
	Output	citra_terskala batas
	Logika proses	1. Jika user memilih menu penormalisasian, maka citra karakter akan dinormalisasi dengan cara mencari batas citra karakter dan penskalaan terhadap citra karakter. 2. Citra tersegmen dicari batas atas, kiri, kanan, dan bawahnya. 3. Batas-batas yang sudah didapatkan digunakan untuk proses penskalaan dengan menggunakan persamaan 2-4 dan persamaan 2-5.
	Keterangan proses	Penormalisasian dilakukan untuk menyesuaikan ukuran citra karakter yang akan dikenali dengan citra karakter *template*.
5.	No. Proses	5
	Nama Proses	Pengenalan

No.	Proses	Keterangan
	Input	batas citra_terskala
	Output	hasil_pengenalan_FE hasil_pengenalan_TM waktu_FE waktu_TM
	Logika proses	1. Pengenalan dengan algoritma *template matching*. 2. Pengenalan dengan algoritma *feature extraction*.
	Keterangan proses	Proses pengenalan dilakukan dengan algoritma *template matching* dan *feature extracion*.

3.1.5.3 Data Flow Diagram Level 2 Proses 2 (Preprocessing)

Terdapat dua proses dalam tahap preprocessing yaitu proses *grayscaling* dan binerisasi. Gambar 3.15 adalah DFD level 2 proses *preprocessing*.



Gambar 3.15. DFD Level 2 Proses *Preprocessing*.

Tabel 3.12 adalah spesifikasi proses DFD level 2 proses *preprocessing*.

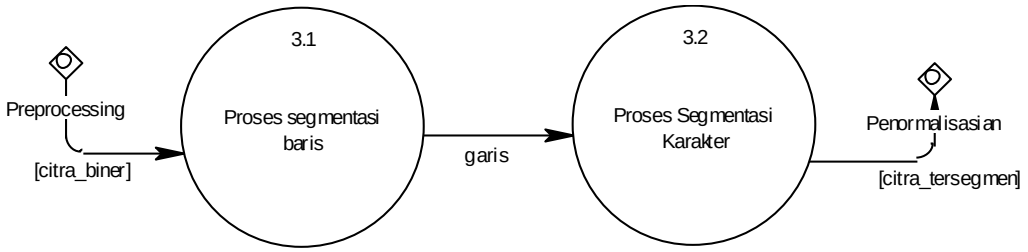
Tabel 3.12. Spesifikasi proses DFD level 2 proses *preprocessing*.

No.	Proses	Keterangan
1.	No. Proses	2.1
	Nama Proses	*Grayscaling*
	Input	citra
	Output	citra_grayscale
	Logika proses	1. Jika menu grayscale dipilih, maka ubah intensitas pixel citra menjadi citra keabuan dengan persamaan 3-1
	Keterangan proses	Perubahan citra dengan intensitas tiga layer menjadi satu layer.
2.	No. Proses	2.2
	Nama Proses	Binerisasi
	Input	citra_grayscale
	Output	citra_biner

No.	Proses	Keterangan
	Logika proses	1. Jika menu binerisasi dipilih, maka bandingkan intensitas pixel citra. Jika pixel citra lebih besar dari 150, maka pixel citra akan berwarna putih dan sebaliknya.
	Keterangan proses	Proses ini dilakukan agar citra hanya memiliki dua warna yaitu hitam atau putih sehingga mudah dalam pemrosesannya.

3.1.5.4 *Data Flow Diagram* Level 2 Proses 3 (Pensegmentasian)

Proses pensegmentasian terbagi menjadi dua proses yaitu proses segmentasi baris dan segmentasi karakter. Segmentasi baris dilakukan untuk memisahkan citra karakter antar baris kalimat. Sedangkan segmentasi karakter dilakukan untuk memisahkan citra karakter yang akan dikenali. Gambar 3.16 adalah DFD level 2 proses pensegmentasian.



Gambar 3.16. DFD Level 2 Proses Pensegmentasian.

Tabel 3.13 adalah tabel yang berisi spesifikasi proses DFD level 2 proses pensegmentasian.

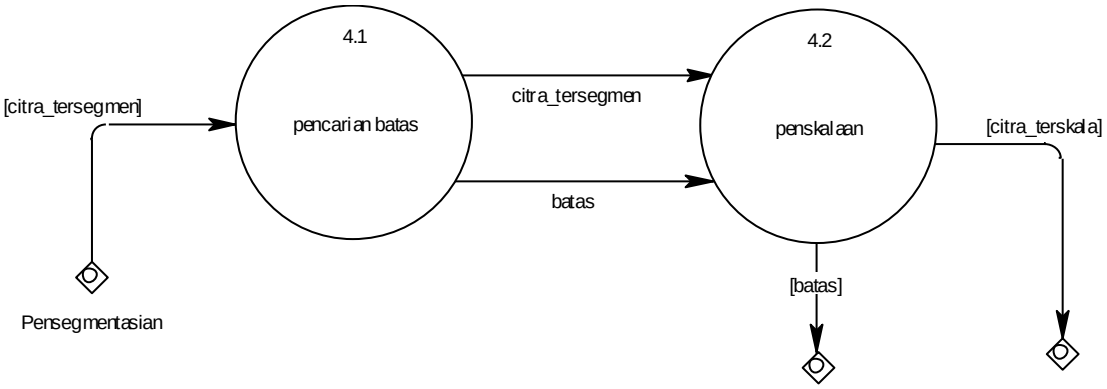
Tabel 3.13. Spesifikasi Proses DFD Level 2 Proses Pensegmentasian.

No.	Proses	Keterangan
1.	No. Proses	3.1
	Nama Proses	Proses segmentasi baris
	Input	citra_biner
	Output	garis
	Logika proses	1. User memilih menu segmentasi baris 2. Pindai pixel citra tiap baris 3. Jika tidak ada pixel hitam pada satu baris pixel, maka tidak ada karakter yang terdeteksi. Jika ada pixel hitam yang terdeteksi pada satu baris pixel, berarti ada karakter yang terdeteksi pada

No.	Proses	Keterangan
		baris tersebut.
	Keterangan proses	Proses ini digunakan untuk memisahkan citra karakter antar baris kalimat.
2.	No. Proses	3.2
	Nama Proses	Proses segmentasi karakter
	Input	Garis
	Output	citra_tersegmen
	Logika proses	1. User memilih menu segmentasi karakter. 2. Pindai pixel citra tiap baris 3. Jika ditemukan pixel berwarna hitam, maka periksa intensitas pixel di sekitarnya. Jika ditemukan pixel hitam disekitarnya, berarti pixel yang sedang dianalisis merupakan satu objek dengan pixel sekitarnya.
	Keterangan proses	Proses ini dilakukan untuk memisahkan karakter yang akan dikenali.

3.1.5.5 *Data Flow Diagram* Level 2 Proses 4 (Penormalisasian)

Penormalisasian terbagi menjadi dua proses yaitu proses pencarian batas dan penskalaan. Proses penskalaan mutlak dilakukan untuk pengenalan dengan metode *template matching*. Pada pengenalan dengan metode *feature extraction*, proses penskalaan dilakukan jika tinggi atau lebar citra lebih dari 60 *pixel*. Hal ini dilakukan karena *array* yang digunakan untuk menganalisis fitur keterbukaan citra berukuran 60 *array*. Gambar 3.17 merupakan DFD level 2 proses penormalisasian.



Gambar 3.17. DFD Level 2 Proses Penormalisasian.

Tabel 3.14 adalah spesifikasi proses DFD level 2 proses normalisasi.

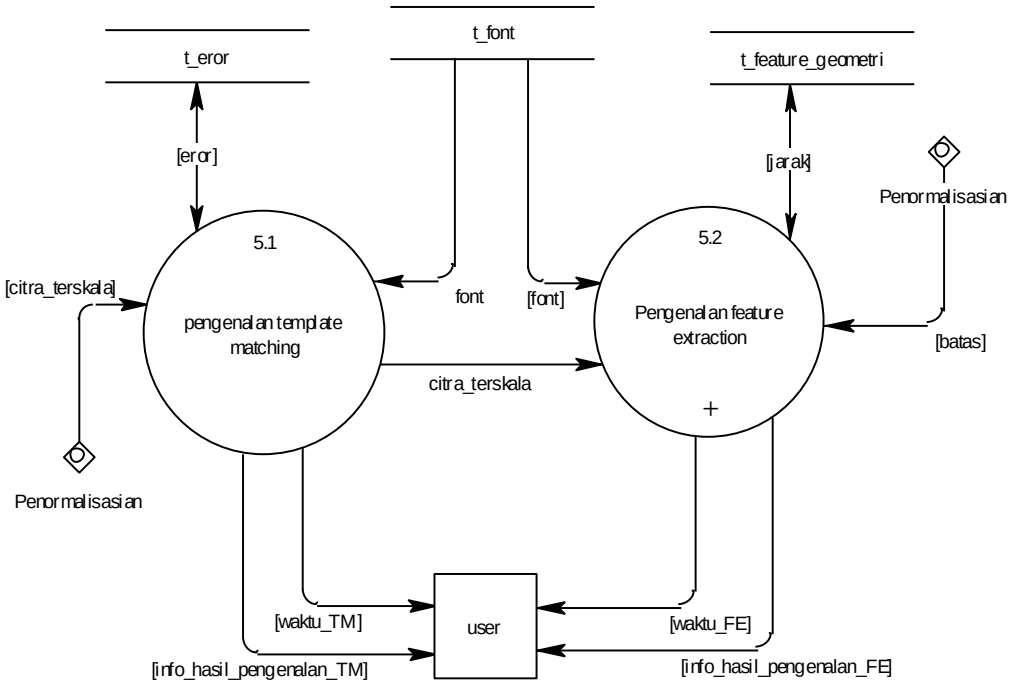
Tabel 3.14. Spesifikasi Proses DFD Level 2 Proses Penormalisasian.

No.	Proses	Keterangan
1.	No. Proses	4.1
	Nama Proses	pencarian batas
	Input	citra_tersegmen
	Output	citra_tersegmen batas
	Logika proses	1. Pixel hitam yang berada paling kiri citra karakter akan menjadi batas kiri. 2. Pixel hitam yang berada paling kanan citra karakter akan menjadi batas kanan. 3. Pixel hitam yang berada paling atas citra karakter akan menjadi batas atas. 4. Pixel hitam yang berada paling bawah citra karakter akan menjadi batas bawah.
	Keterangan proses	Batas-batas karakter berguna dalam proses penskalaan dan analisis karakter
2.	No. Proses	4.2
	Nama Proses	penskalaan
	Input	citra_tersegmen batas
	Output	citra_terskala batas
	Logika proses	1. Penskalaan dilakukan berdasarkan batas-batas yang telah didapatkan pada proses pencarian batas. 2. Penskalaan dilakukan agar citra karakter memiliki lebar 10 pixel dan tinggi 12 pixel. 3. Penskalaan dilakukan dengan menggunakan persamaan 2-4 dan persamaan 2-5.
	Keterangan proses	Proses ini dilakukan agar ukuran citra karakter sesuai dengan ukuran citra *template* pada basis data.

3.1.5.6 *Data Flow Diagram* Level 2 Proses 5 (Pengenalan)

Proses pengenalan terbagi menjadi dua proses yaitu proses *template matching* dan *feature extraction*. Masing-masing proses ini akan mengenali citra sesuai dengan metodenya. Proses pertama yang dilakukan adalah pengenalan dengan metode *template matching* dan selanjutnya dokumen dikenali dengan *feature extraction*. Hasil keluaran dari masing-masing proses ini adalah hasil

pengenalan dan waktu yang dibutuhkan masing-masing algoritma dalam mengenali citra karakter. Gambar 3.18 adalah DFD level 2 proses pengenalan.



Gambar 3.18. DFD Level 2 Proses Pengenalan.

Tabel 3.15 adalah spesifikasi proses DFD level 2 proses pengenalan.

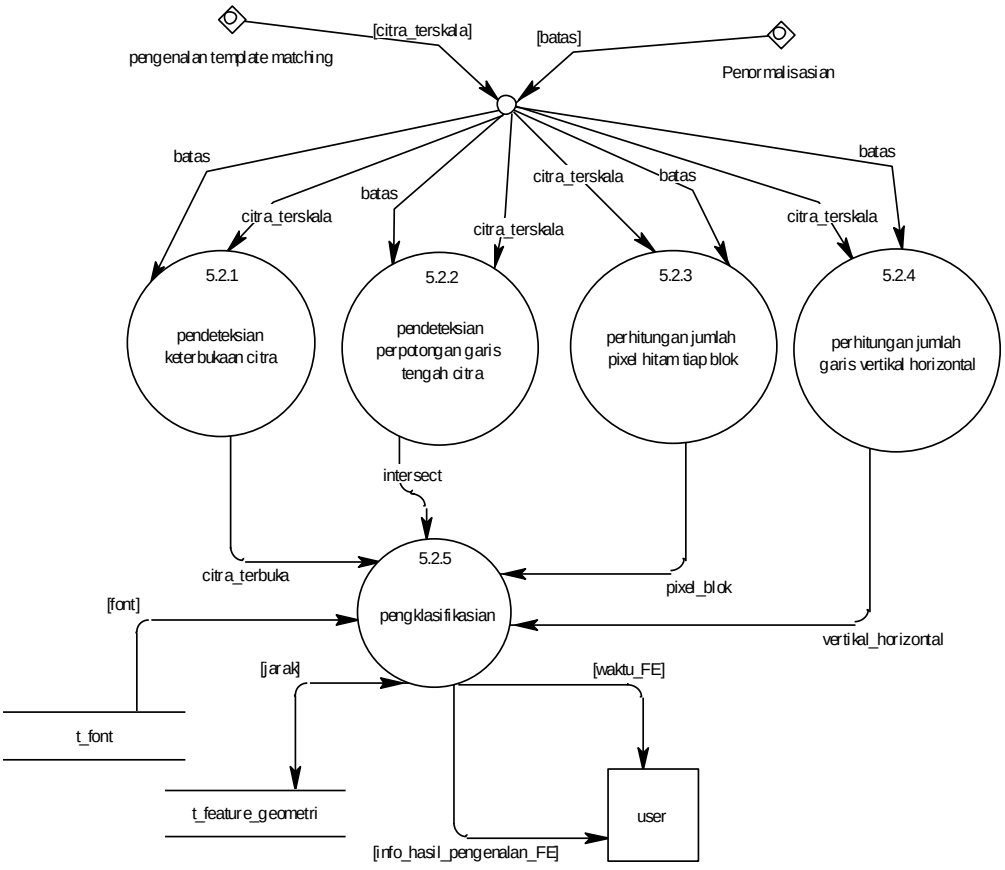
Tabel 3.15. Spesifikasi Proses DFD Level 2 Proses Pengenalan.

No.	Proses	Keterangan
1.	No. Proses	5.1
	Nama Proses	*Template matching*
	Input	citra_terskala
	Output	hasil_pengenalan_TM waktu_TM
	Logika proses	1. Jika user memilih menu Template Matching, citra ternormalisasi akan dibandingkan dengan citra pada *template* di basis data dengan menghitung nilai eror. 2. Nilai eror terkecil akan merepresentasikan *template* yang paling sesuai dengan citra input.
	Keterangan proses	Proses ini merupakan pengenalan dengan metode *template matching*
2.	No. Proses	4.2
	Nama Proses	*Feature extraction*
	Input	citra_terskala
	Output	hasil_pengenalan_FE waktu_FE

No.	Proses	Keterangan
	Logika proses	1. Jika user memilih menu Feature Extraction, citra normalisasai akan diambil ciri-cirinya sesuai dengan atribut citra *template* pada basis data. 2. Ciri-ciri citra input akan dikenali dengan cara mengklasifikasikannya terhadap citra *template* pada basis data.
	Keterangan proses	Proses ini merupakan proses pengenalan dengan metode *feature extraction*.

3.1.5.7 Data Flow Diagram Level 3 Proses 4.2 (*Feature extraction*)

Pada proses pengenalan dengan metode *feature extraction*, citra diambil ciri-cirinya terlebih dahulu sebelum diklasifikasikan. Gambar 3.19 adalah DFD level 3 proses pengenalan.



Gambar 3.19. DFD Level 3 Proses *Feature Extraction*.

Tabel 3.16 adalah spesifikasi proses DFD level 3 proses *feature extraction*.

Tabel 3.16. Spesifikasi Proses DFD level 3 Proses *Feature Extraction*.

No.	Proses	Keterangan
1.	No. Proses	5.2.1
	Nama Proses	pendeteksian keterbukaan citra
	Input	citra_terskala batas
	Output	citra_terbuka
	Logika proses	1. Citra karakter dideteksi apakah terbuka ke atas, bawah, kiri, atau kanan sesuai analasisi pada sub bab 3.1.3.6 yang membahas ciri citra karakter terbuka.
2.	No. Proses	5.2.2
	Nama Proses	pendeteksian perpotongan garis tengah citra
	Input	citra_terskala batas
	Output	Intersect
	Logika proses	1. Garis tengah vertikal dihitung dengan cara menjumlahkan batas kanan dan kiri kemudian dibagi dua. 2. Garis tengah horizontal dihitung dengan cara menjumlahkan batas kanan dan kiri kemudian dibagi dua. 3. Telusuri pixel pada masing-masing garis. 4. Jika ditemukan pola pixel hitam dan putih berturut-turut, maka akan dihitung sebagai satu potongan.
3.	No. Proses	5.2.3
	Nama Proses	perhitungan jumlah pixel hitam tiap blok
	Input	citra_terskala batas
	Output	vertikal_horizontal
	Logika proses	1. Bagi citra karakter menjadi sembilan blok simetris dengan menggunakan batas-batas citra karakter tersebut. 2. Hitung jumlah pixel hitam pada masing-masing blok yang dibagi dengan resolusi citra karakter.
4.	No. Proses	5.2.4
	Nama Proses	perhitungan jumlah garis vertikal dan horizontal
	Input	citra_terskala batas
	Output	*Pixel*_blok
	Logika proses	1. Citra karakter dideteksi jumlah garis vertikal dan horizontalnya sesuai dengan sub bab 3.1.3.4 dan sub bab 3.1.3.5 yang membahas cara menghitung garis vertikal dan horizontal citra karakter.

No.	Proses	Keterangan
6.	No. Proses	5.2.5
	Nama Proses	Pengklasifikasian
	Input	citra_terbuka intersect pixel_blok vertikal_horizontal
	Output	waktu_FE hasil_pengenalan_FE jarak
	Logika proses	1. Ciri-ciri yang dipetakan diklasifikan dengan ciri-ciri yang ada di database dan dihitung jaraknya dengan menggunakan persamaan 2-7. 2. Urutkan jarak secara *ascending*. Jarak terkecil adalah karakter yang paling sesuai dengan citra karakter yang akan dikenali.

3.1.5.8 Kamus Data

Kamus data digunakan untuk menjelaskan semua data yang digunakan di dalam sistem. Tabel 3.17 adalah kamus data berdasarkan sistem yang dibangun.

Tabel 3.17. Kamus Data pada aplikasi perbandingan algoritma *template matching* dan *feature extraction* pada OCR.

No.	Data	Keterangan
1.	Nama	Citra
	Struktur data	Gambar digital berformat .JPG atau .BMP
2.	Nama	info_hasil_pengenalan_FE
	Struktur data	info_hasil_pengenalan_FE
	Deskripsi	info_hasil_pengenalan_FE= [a..z]+[0..9]
3.	Nama	waktu_FE
	Struktur data	waktu_FE
	Deskripsi	waktu_FE = [a..z]+[0..9]
4.	Nama	info_hasil_pengenalan_TM
	Struktur data	info_hasil_pengenalan_TM
	Deskripsi	hasil_pengenalan_TM =[a..z]
5.	Nama	waktu_TM
	Struktur data	waktu_TM
	Deskripsi	waktu_TM = [a..z]+[0..9]
6.	Nama	citra_tersegmen
	Struktur data	citra_tersegmen
	Deskripsi	citra yang masing-masing citra karakternya terpisah dengan karakter lainnya. Terpisahnya karakter diwakili oleh batas-batas karakter.
7.	Nama	citra_terskala
	Struktur data	citra_terskala

No.	Data	Keterangan
	Deskripsi	Citra karakter yang diskalakan dengan resolusi 10x12 pixel.
8.	Nama	Batas
	Struktur data	atas+bawah+kiri+kanan
	Deskripsi	atas=[0..9] bawah=[0..9] kiri=[0..9] kanan=[0..9]
9.	Nama	citra_grayscale
	Struktur data	citra_grayscale
	Deskripsi	Citra dengan warna keabuan
10.	Nama	citra_biner
	Struktur data	citra_biner
	Deskripsi	Citra dengan warna hitam atau putih
11.	Nama	jarak
	Struktur data	jarak
	Deskripsi	jarak=[0..9]
12.	Nama	Eror
	Struktur data	Eror
	Deskripsi	eror=[0..9]
13.	Nama	Keterbukaan
	Struktur data	t_atas+t_bawah+t_kanan+t_kiri
	Deksipsi	t_atas=[0|1] t_bawah=[0|1] t_kanan=[0|1] t_kiri=[0|1]
14.	Nama	intersect
	Struktur data	Interv+interh
	Deskripsi	interv=[0..9] interh=[0..9]
15.	Nama	pixel_blok
	Struktur data	blok1+blok2+blok3+blok4+blok5+blok6+blok7+blok8+blok9
	Deskripsi	blok1=[0..9] blok2=[0..9] blok3=[0..9] blok4=[0..9] blok5=[0..9] blok6=[0..9] blok7=[0..9] blok8=[0..9] blok9=[0..9]
16.	Nama	vertikal_horizontal
	Struktur data	vertikal+horizontal
	Deskripsi	vertikal=[0..9] horizontal=[0..9]
17.	Nama	garis
	Struktur data	garis[0..tinggi_citra-1]
	Deskripsi	garis[0..tinggi_citra-1] =[0..9]
18.	Nama	data_permintaan_citra

No.	Data	Keterangan
	Deskripsi	Data ini mengalir akibat user memilih menu open
19.	Nama	font
	Struktur data	id_font
	Deskripsi	id_font=[A|T]+[0..9]

3.2 Perancangan Sistem

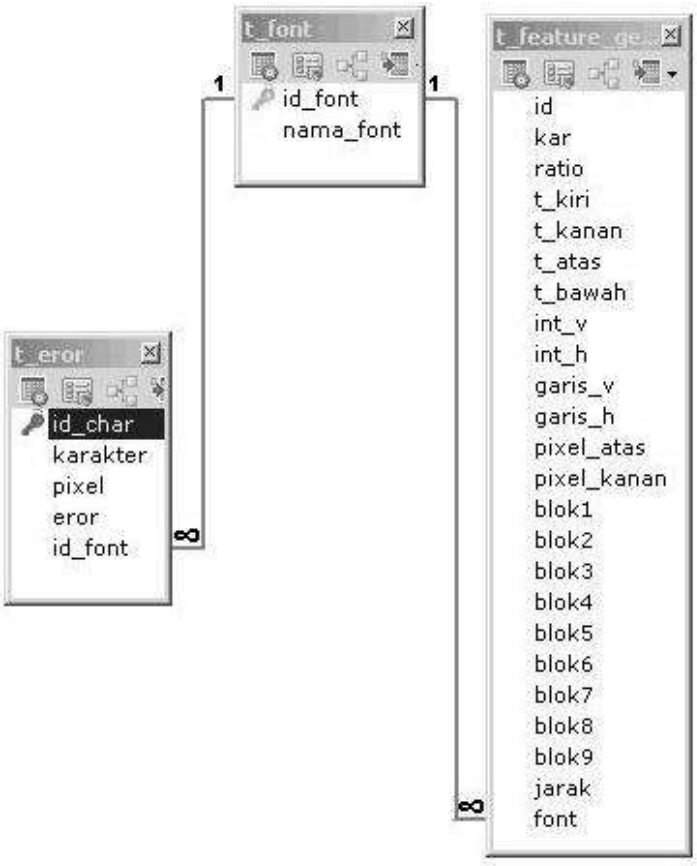
Perancangan merupakan tahap lanjutan dari analisis sistem dimana pada perancangan sistem digambarkan rancangan sistem yang akan dibangun sebelum dilakukan pengkodean kedalam suatu bahasa pemrograman.

3.2.1 Perancangan Data

Perancangan data terdiri dari perancangan skema relasi berdasarkan analisis basis data dan struktur tabel yang dibangun dalam sistem.

3.2.1.1 Skema Relasi

Gambar 3.20 adalah skema relasi basis data yang dibangun pada aplikasi perbandingan algoritma *template matching* dan *feature extraction* pada OCR.



Gambar 3.20. Skema Relasi Aplikasi Perbandingan Algoritma *Template Matching* dan *Feature Extraction* pada OCR.

3.2.1.2 Strukur Tabel

Tabel 3.18, tabel 3.19, tabel 3.20 adalah struktur tabel berdasarkan skema relasi pada sistem.

Tabel 3.18. Struktur Tabel t_eror.

Nama Field	Tipe	Panjang	Keterangan
id_char	varchar	8	primary_key
karakter	char	1	
pixel	varchar	150	
eror	int	11	
id_*font*	varchar	5	foreign_key

Tabel 3.19. Struktur Tabel t_*font*.

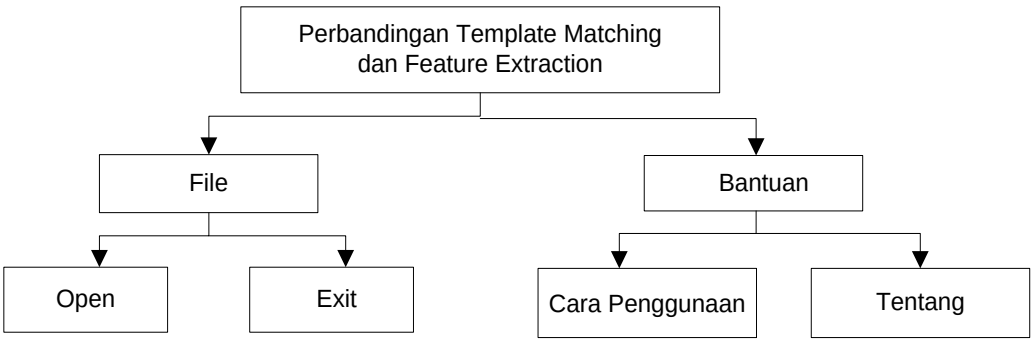
Nama Field	Tipe	Panjang	Keterangan
id_*font*	varchar(5)	5	primary_key
nama_*font*	varchar(50)	50	

Tabel 3.20. Struktur Tabel t_*feature*_geometri.

Nama Field	Tipe	Panjang	keterangan
id	int	11	primary_key
kar	varchar	3	
ratio	float		
t_kiri	int	11	
t_kanan	int	11	
t_atas	int	11	
t_bawah	int	11	
int_v	int	11	
int_h	int	11	
garis_v	int	11	
garis_h	int	11	
blok1	float		
blok2	float		
blok3	float		
blok4	float		
blok5	float		
blok6	float		
blok7	float		
blok8	float		
blok9	float		
jarak	float		
Font	char	1	

3.2.2 Perancangan Struktur Menu

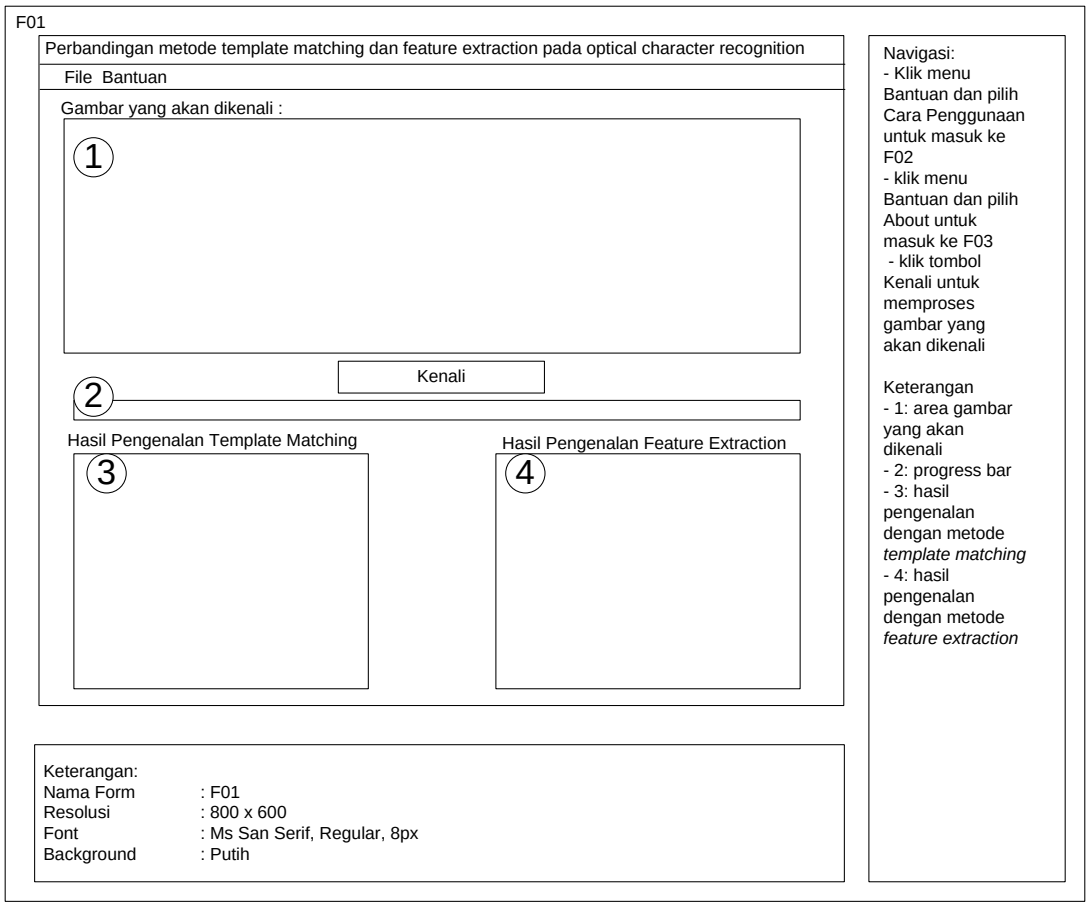
Gambar 3.21 adalah perancangan struktur menu pada sistem yang dirancang.



Gambar 3.21. Struktur Menu.

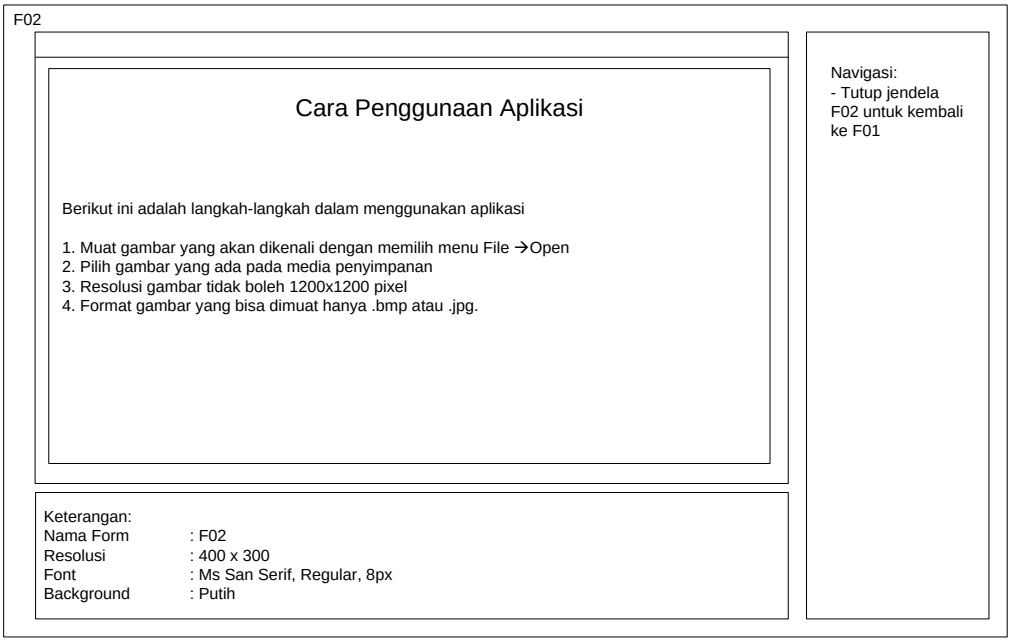
3.2.3 Perancangan Antarmuka

Gambar 3.22 adalah gambar perancangan antarmuka form utama.



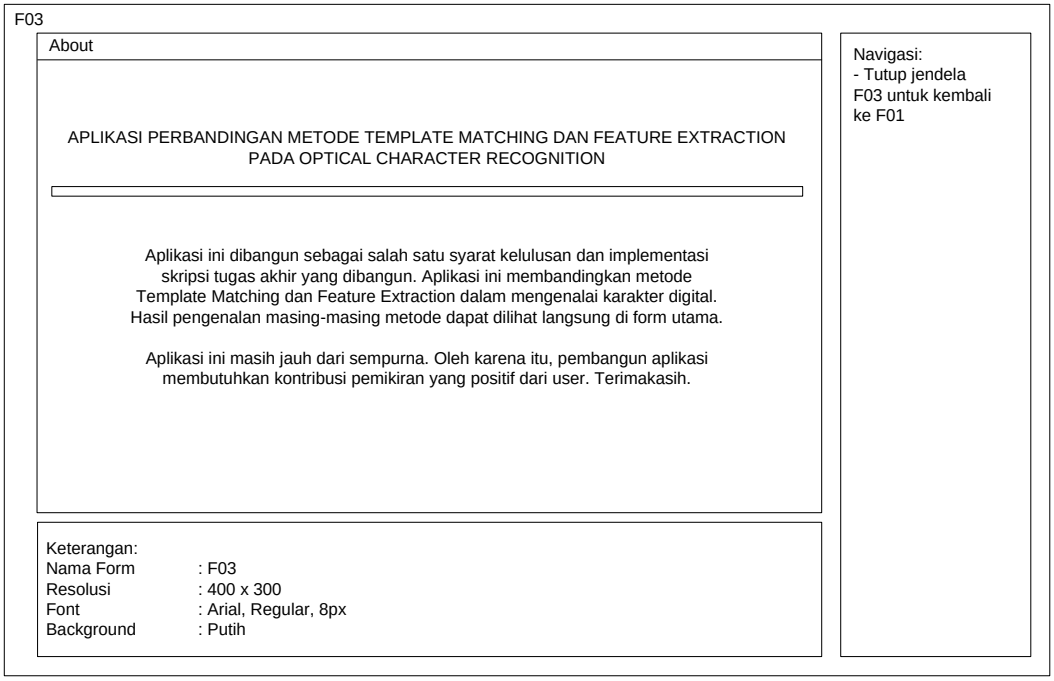
Gambar 3.22. Tampilan *Form* Utama.

Gambar 3.23 adalah perancangan antarmuka form cara penggunaan.



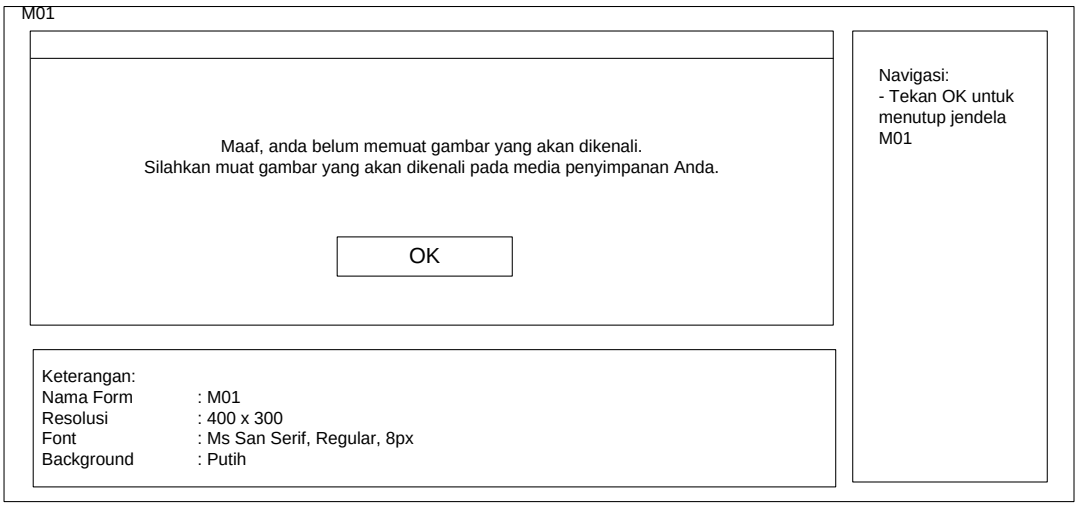
Gambar 3.23. Tampilan *Form* Cara Penggunaan.

Gambar 3.24 adalah perancangan antarmuka form tentang aplikasi.



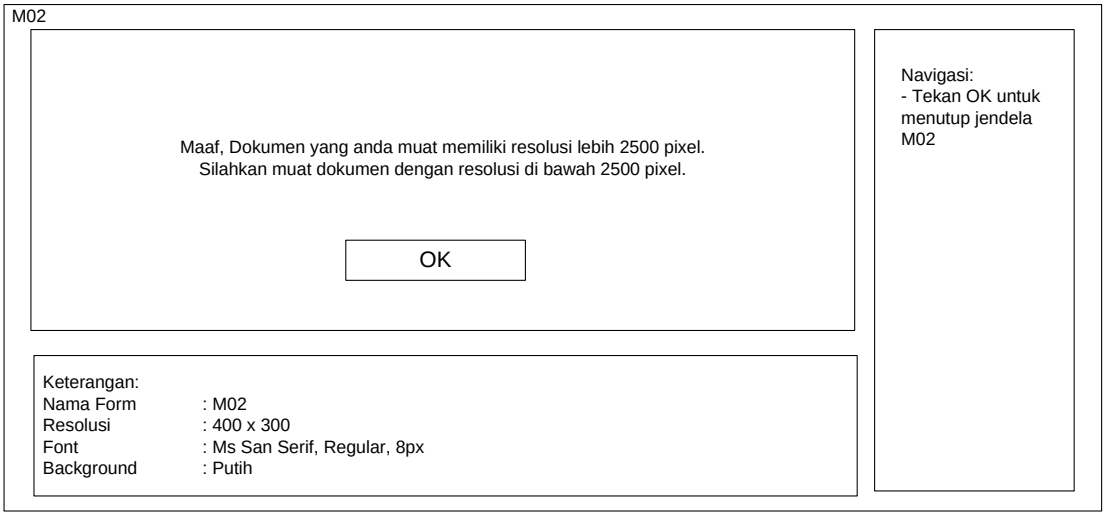
Gambar 3.24. Tampilan *Form* About.

Gambar 3.25 adalah perancangan pesan eror yang muncul jika user menekan tombol Kenali namun dokumen yang akan dikenali belum dimuat.



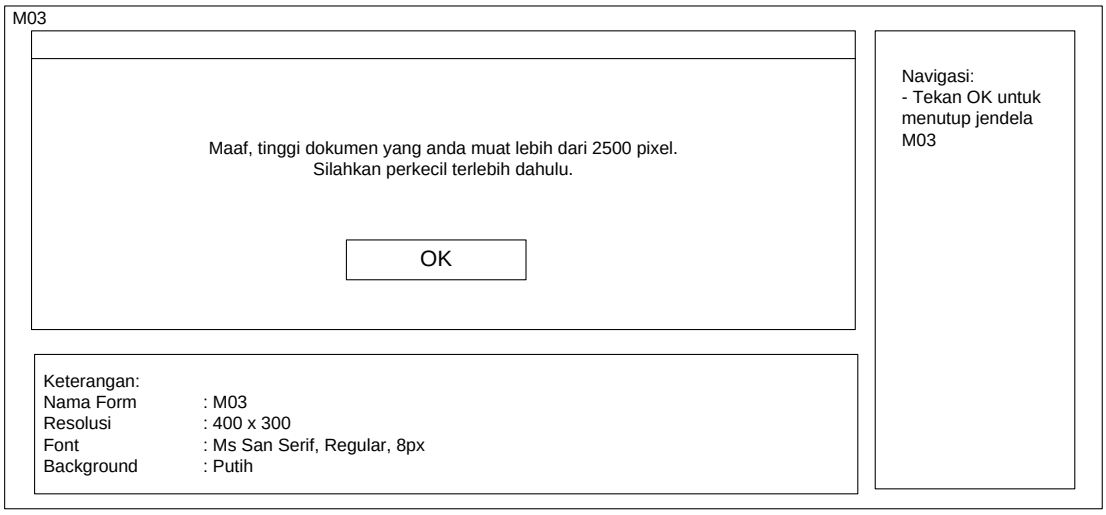
Gambar 3.25. Tampilan Pesan Kesalahan M01.

Gambar 3.26 adalah perancangan pesan eror jika dokumen yang dimuat memiliki resolusi lebih dari 2500x2500 *pixel*.



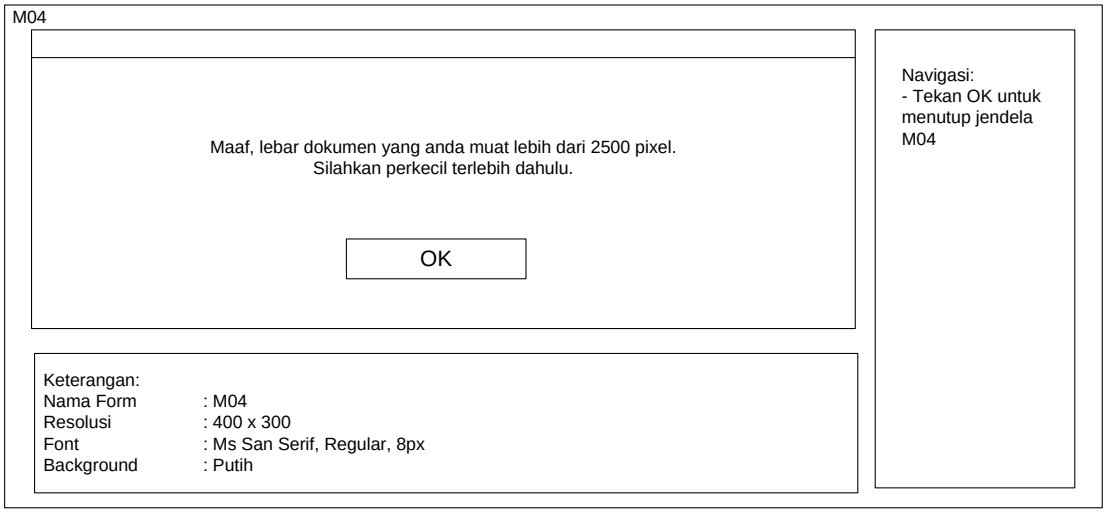
Gambar 3.26. Tampilan Pesan Kesalahan M02.

Gambar 3.27 adalah perancangan pesan kesalahan jika dokumen yang dimuat memiliki tinggi lebih dari 2500 *pixel*.



Gambar 3.27. Tampilan Pesan Kesalahan M03.

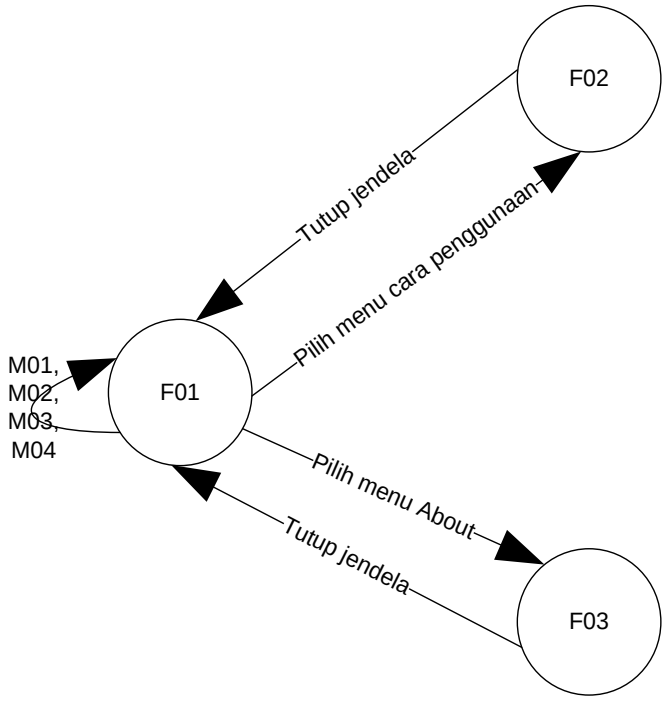
Gambar 3.28 adalah perancangan pesan kesalahan jika dokumen yang dimuat memiliki lebar lebih dari 2500 *pixel*.



Gambar 3.28. Tampilan Pesan Kesalahan M04.

3.2.4 Jaringan Semantik

Gambar 3.29 adalah jaringan semantik yang terbentuk berdasarkan perancangan antarmuka.



Gambar 3.29. Jaringan Semantik.

3.2.5 Perancangan Prosedural

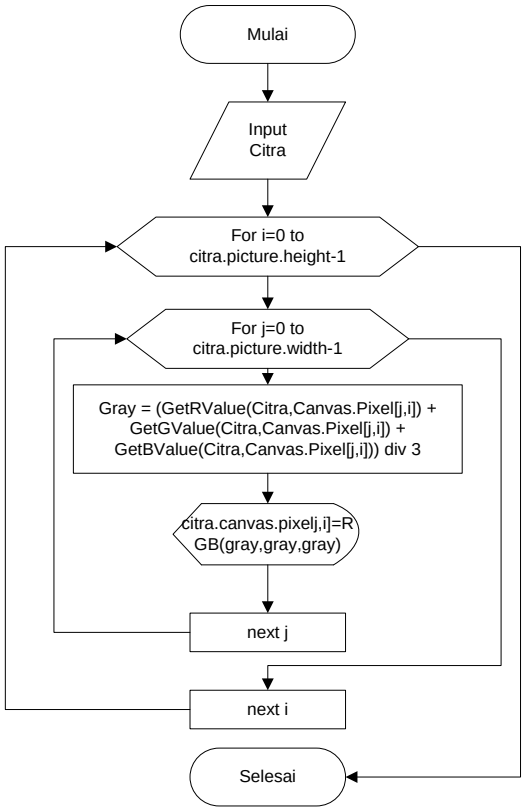
Perancangan prosedural merupakan perancangan yang dilakukan untuk menetapkan detail algoritma yang akan dinyatakan ke dalam suatu perangkat lunak.

3.2.5.1 Tahap Preprocessing

Citra digital yang dimuat memiliki intensitas *pixel* yang beragam. Tahap preprocessing berguna untuk membatasi intensitas *pixel* citra.

3.2.5.1.1 Grayscaling Citra

Gambar 3.30 adalah *flowchart* algoritma yang digunakan untuk *grayscaling* citra.



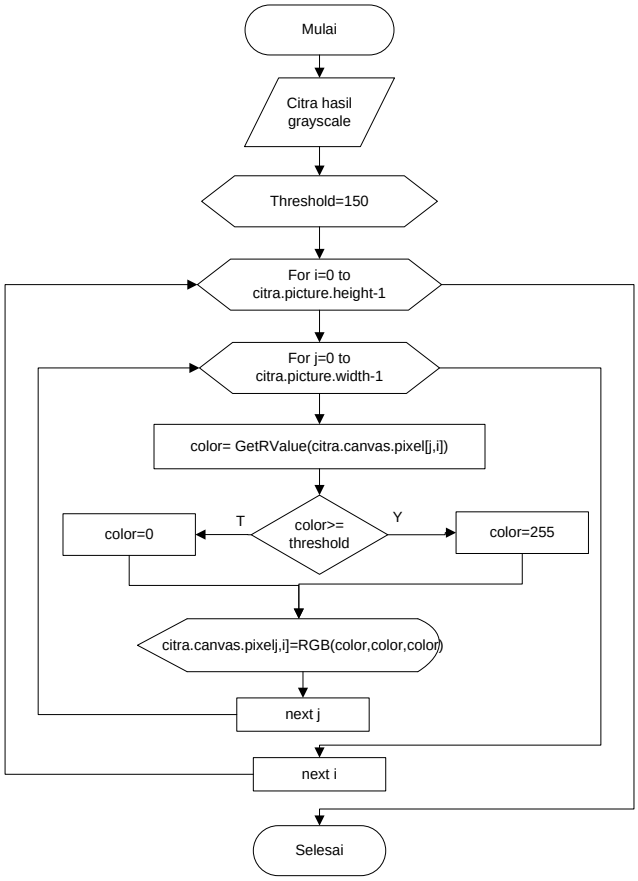
Gambar 3.30. *Flowchart* Proses *Grayscaling*.

Keterangan gambar 3.30:

1. citra.picture.height : tinggi citra
2. citra.picture.width : lebar citra
3. citra.canvas.pixel[j,i] : fungsi yang digunakan untuk mengakses intensitasa pixel dengan koodinat j,i.
4. GetRValue : fungsi yang digunakan untuk mendapatkan nilai *layer* R (*red*) suatu pixel
5. GetGValue : fungsi yang digunakan untuk mendapatkan nilai *layer* G (*green*) suatu pixel.
6. GetBValue : fungsi yang digunakan untuk mendapatkan nilai *layer* B (*blue*) suatu pixel.
7. RGB(Gray, Gray, Gray) : fungsi yang digunakan untuk menggabungkan ketiga *layer* menjadi satu *layer*.

3.2.5.1.2 Proses Binerisasi Citra

Gambar 3.31 adalah *flowchart* algoritma yang digunakan pada proses binerisasi.



Gambar 3.31. *Flowchart* Proses Binerisasi.

Keterangan gambar 3.31:

1. Threshold : nilai yang menjadi batas intensitas pixel. Jika suatu intensitas pixel kurang dari nilai ini, maka akan berwarna hitam. Jika lebih dari nilai threshlod, maka akan berwarna putih.
2. citra.picture.height : tinggi citra
3. citra.picture.width : lebar citra
4. citra.canvas.pixel[j,i]=clBlack: apakah pixel yang sedang diakses berwarna hitam
5. GetRValue : fungsi yang digunakan untuk mendapatkan nilai *layer* R (*red*) suatu pixel.

6. RGB(Color,Color,Color) : fungsi yang digunakan untuk menggabungkan ketiga *layer* (Color,Color,Color) menjadi satu *layer*.

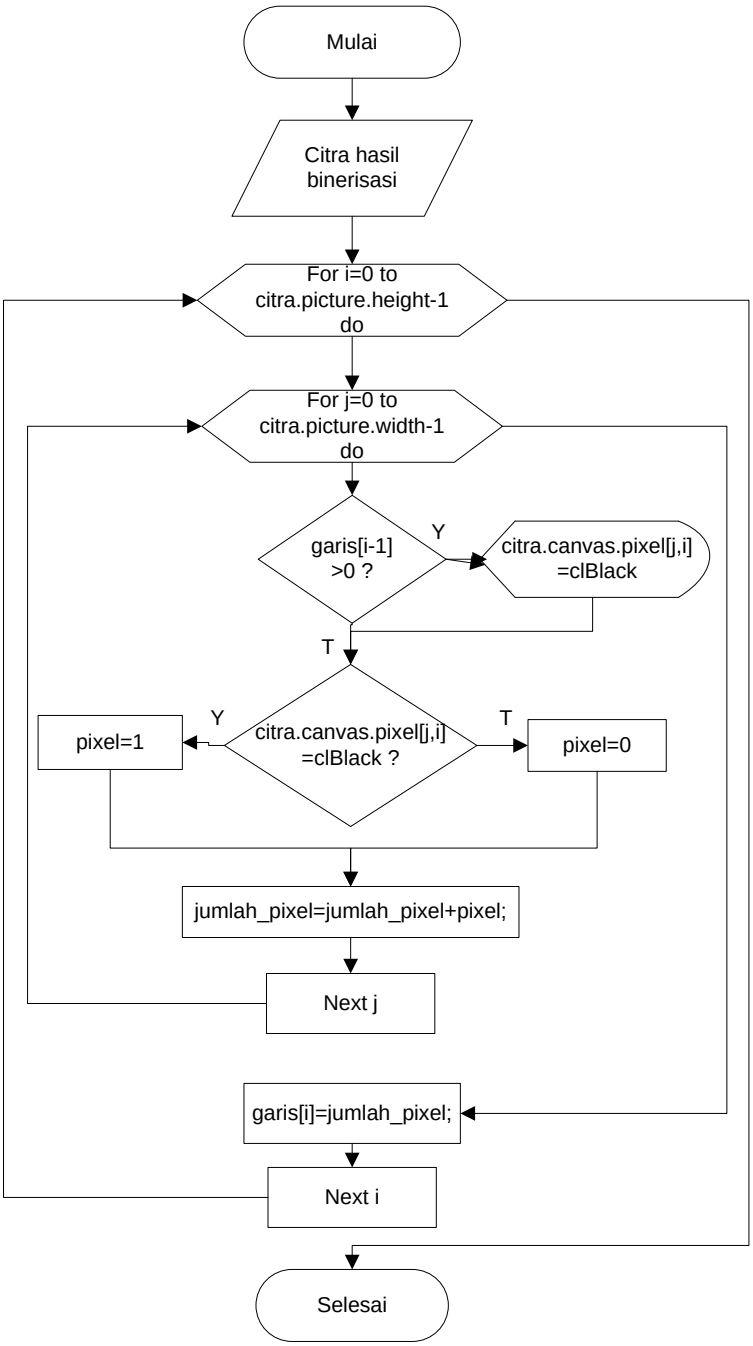
Berdasarkan *flowchart* pada gambar 3.31, jika nilai *pixel* lebih besar dari 150 maka *pixel* akan direprentasikan dengan warna putih dan sebaliknya. Nilai 150 didapatkan dari hasil percobaan dan merupakan nilai terbaik untuk binerisasi citra yang berpengaruh terhadap akurasi pada proses pengenalan.

3.2.5.2 Tahap Pensegmentasian

Citra digital hasil *preprocessing* akan disegmentasi untuk memisahkan karakter-karakter yang akan dikenali dari latar belakang citra. Proses yang pertama dilakukan adalah segmentasi baris, kemudian segmentasi karakter.

3.2.5.2.1 Segmentasi Baris

Gambar 3.32 adalah *flowchart* algoritma yang digunakan untuk proses segmentasi baris.



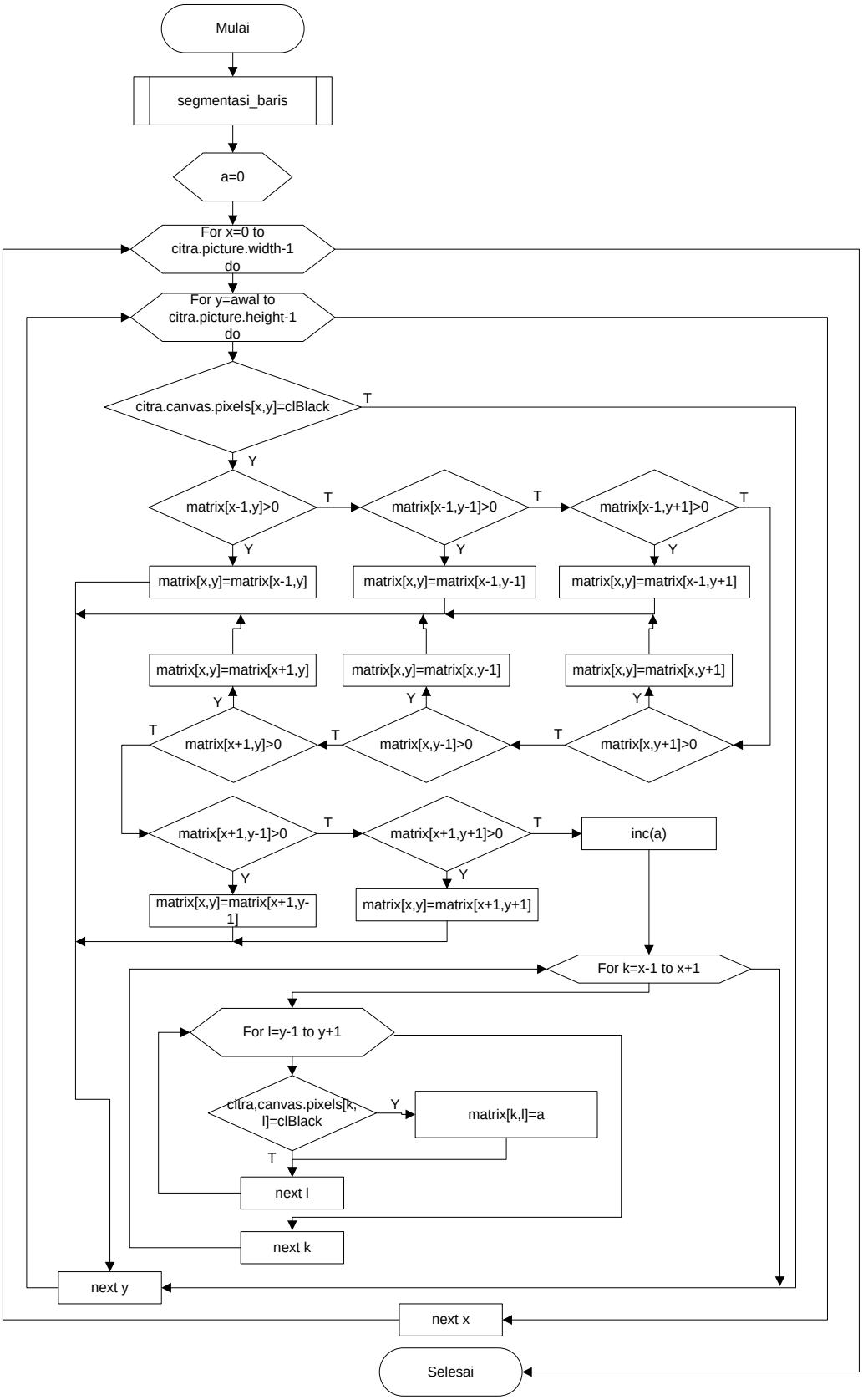
Gambar 3.32. *Flowchart* Segmentasi Baris.

Keterangan gambar 3.32:

1. citra.picture.height : tinggi citra
2. citra.picture.width : lebar citra
3. citra.canvas.pixel[j,i]=clBlack?: apakah pixel yang sedang diakses berwarna hitam ?

3.2.5.2.2 Segmentasi Karakter

Gambar 3.33 adalah flowchart algoritma yang digunakan untuk segmentasi karakter dengan menggunakan metode *Connected Component Analyisis*.



Gambar 3.33. *Flowchart* Algoritma *Connected Component Analysis*.

Keterangan gambar 3.33:

1. citra.picture.height-1 : tinggi citra -1
2. citra.picture.width-1 : lebar citra – 1
3. inc(a) : inc adalah fungsi *increment* yang berguna untuk menambah jumlah a. a adalah variabel yang digunakan untuk menyimpan jumlah karakter.
4. citra.canvas.pixel[j,i]=clBlack : pixel yang sedang diakses berwarna hitam
5. matrix[x-1,y]>0 : [x-1, y] adalah koordinat pixel tetangga di sebelah kiri atas pixel yang sedang dianalisis. Jika matrix[x-1,y]>0 berarti sudah ada label pada pixel dengan posisi tersebut. Hal yang sama dilakukan pada matrix pada posisi lainnya.

Berdsarkan *flowchart* pada gambar 3.33, jika ditemukan pixel berwarna hitam, maka periksa pixel tetangga yang berada di kiri atas. Jika pixel pada posisi tersebut berwarna hitam dan telah memiliki label, maka pixel yang sedang di analisis terhadap tetangga-tetangganya akan diberi label yang sama dan dianggap sebagai satu objek dengan pixel yang berada pada posisi kiri atas. Cara yang sama juga dilakukan terhadap tujuh posisi pixel tetangga lainnya.

Jika tidak ada pixel yang berwarna hitam pada seluruh tetangga, maka beri label baru terhadap pixel yang sedang dianalisis. Setelah diberi label baru, periksa kedelapan pixel tetangga. Jika ditemukan pixel berwarna hitam, maka beri label yang sama dengan pixel yang sedang dianalisis. Label diberikan dengan cara mencatat jumlah objek pada array dua dimensi.

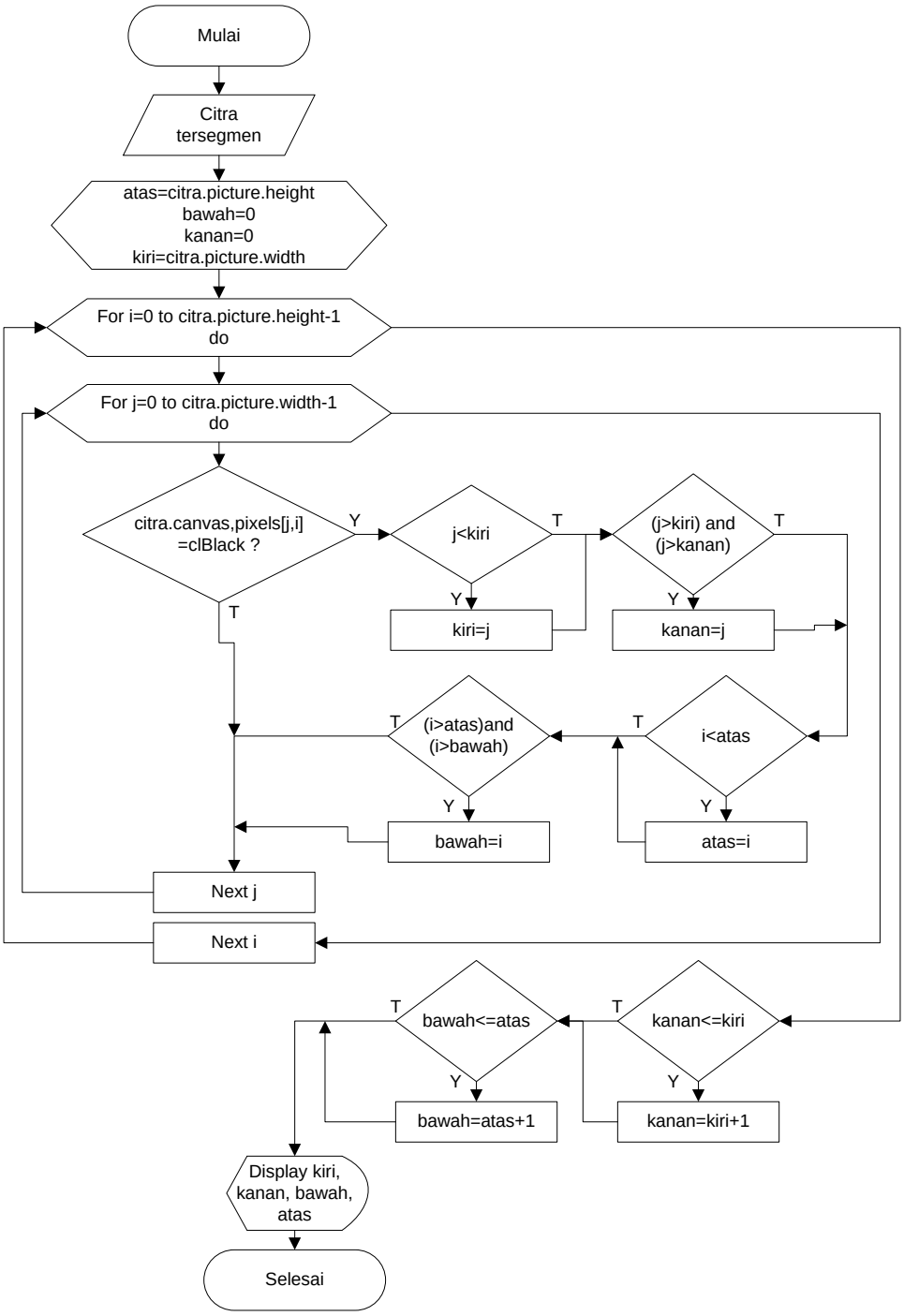
3.2.5.3 Tahap Penormalisasian

Ada dua tahap dalam proses penormalisasian yaitu tahap pencarian batas dan penskalaan. Pencarian batas berguna untuk proses penskalaan dan untuk

pengenalan dengan algoritma *feature extraction*. Penskalaan mutlak digunakan untuk proses pengenalan dengan metode *template matching*. Penskalaan juga dilakukan untuk pengenalan dengan metode *feature extraction* jika tinggi atau lebar citra karakter lebih dari 60 pixel. Penskalaan dilakukan untuk menyesuaikan resolusi *pixel* citra masukan dengan citra *template*.

3.2.5.3.1 Tahap Pencarian Batas

Gambar 3.34 adalah flowchart yang digunakan untuk mendeteksi batas-batas citra karakter.



Gambar 3.34. *Flowchart* Pencarian Batas.

Keterangan gambar 3.34:

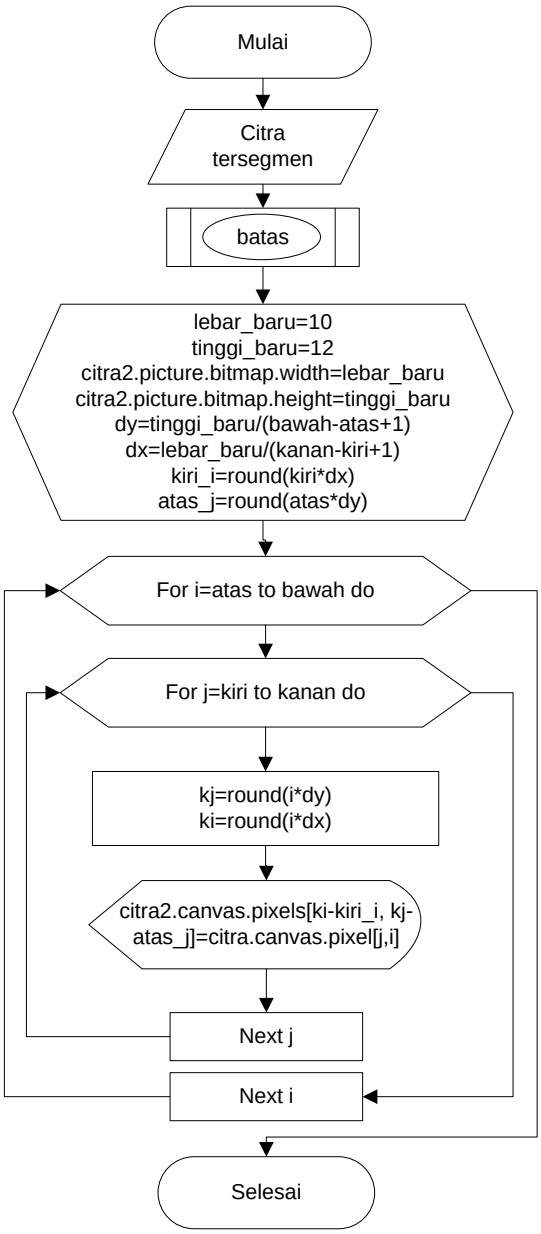
1. citra.picture.height : tinggi citra
2. citra.picture.width : lebar citra

3. kiri, kanan, atas, dan bawah: variabel yang digunakan untuk menyimpan nilai batas-batas citra karakter.
4. citra.canvas.pixel[j,i]=clBlack? : apakah pixel yang sedang diakses berwarna hitam

Berdasarkan *flowchart* pada gambar 3.34, jika ditemukan pixel berwarna hitam, maka periksa koordinatnya. Jika koordinat y-nya lebih kecil dari batas atas, maka koordinat tersebut akan menjadi batas atas. Jika koordinat y lebih besar dari batas bawah, maka koordinat y tersebut akan menjadi batas bawah. Jika koordinat x lebih kecil dari batas kiri, maka koordinat tersebut akan menjadi batas kiri. Jika koordinat x lebih besar dari batas kanan, maka koordinat tersebut akan menjadi batas kanan.

3.2.5.3.2 Tahap Pensklaan

Gambar 3.35 adalah *flowchart* algoritma yang digunakan untuk penskalaan.



Gambar 3.35. *Flowchart* Algoritma Penskalaan.

Keterangan gambar 3.35:

1. citra.picture.height : tinggi citra
2. citra.picture.width : lebar citra
3. citra.picture.bitmap.height : fungsi yang digunakan untuk mengakses resolusi pixel tinggi citra.
4. citra.picture.bitmap.width : fungsi yang digunakan untuk mengakses resolusi pixel tinggi citra.

5. kj,ki : variabel yang digunakan untuk menentukan koordinat baru. Koordinat ini berguna untuk pemetaan pixel pada citra yang belum diskalakan (citra 1) terhadap citra yang diskalakan (citra 2).
6. citra2.canvas.pixel[j,i] : fungsi yang digunakan untuk mengakses intensitas pixel citra2 pada koordinat j,i.

Citra2 digunakan sebagai media untuk menampung citra hasil penskalaan. Citra2 memiliki resolusi pixel dengan lebar dan tinggi citra yang sesuai dengan resolusi citra *template*.

3.2.5.4 Tahap Pengenalan

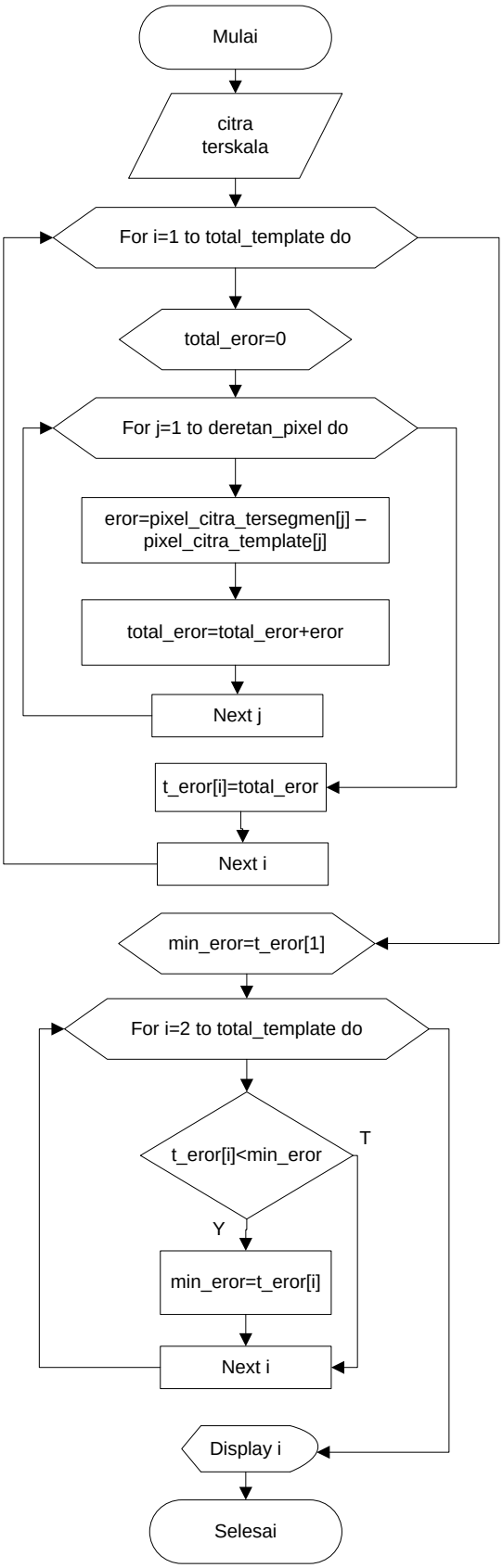
Proses pengenalan dilakukan untuk mengenali citra karakter masukan dengan citra karakter *template*.

3.2.5.4.1 Pengenalan dengan Algoritma *Template matching*

Metode *template matching* mengenali karakter dengan menghitung nilai eror antara deretan angka biner yang mewakili nilai *pixel* citra masukan dan deretan angka biner yang mewakili citra *template*.

Pixel citra karakter masukan yang berwarna hitam akan diwakili dengan angka 1. Sedangkan *pixel* yang berwarna putih akan diwakili dengan angka 0. Nilai eror akan dihitung berdasarkan deretan angka 0 dan 1 antara citra masukan dan citra *template*. Nilai eror terkecil menyatakan *template* yang paling sesuai dengan citra masukan.

Gambar 3.36 adalah *flowchart* algoritma yang digunakan untuk menghitung nilai eror dengan menggunakan persamaan 2-2. Jika nilai eror sudah dihitung, maka cari nilai eror terkecil.



Gambar 3.36. *Flowchart* Perhitungan Nilai Eror.

Keterangan gambar 3.36:

1. pixel_citra_tersegmen : intensitas pixel citra yang akan dibandingkan. Jika berwarna hitam akan diwakili dengan angka 1, jika berwarna putih akan diwakili dengan angka 0.
2. pixel_citra_*template* : intensitas pixel citra *template*. Jika berwarna hitam akan diwakili dengan angka 1, jika berwarna putih akan diwakili dengan angka 0.
3. eror= hasil perhitungan antara pixel citra yang akan dibandingkan dan pixel citra *template*.
4. t_eror[i] : array yang digunakan untuk menyimpan nliai eror masing-masing *template*.
5. t_eror[i] : array yang digunakan untuk menyimpan nliai eror masing-masing *template*.
6. min_eror: variabel yang digunakan untuk menyimpan nilai eror terkecil

Nilai eror terkecil adalah karakter *template* yang paling sesuai dengan citra masukan yang dibandingkan.

3.2.5.4.2 Pengenalan dengan Algoritma *Feature Extraction*

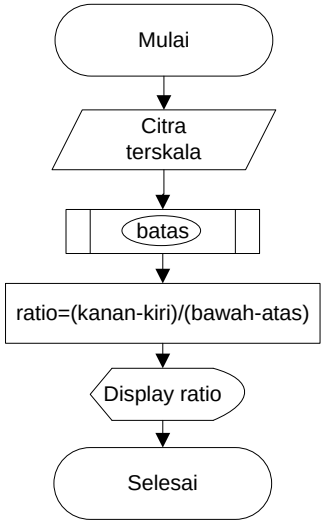
Metode *feature extraction* mengenali karakter dengan menghitung jarak antara atribut-atribut yang dimiliki oleh citra masukan dan citra *template*. Jarak berupa selisih nilai atribut yang diurutkan dari kecil ke besar antara citra masukan dan citra *template*. Citra *template* dengan jarak terkecil adalah *template* yang paling sesuai dengan citra masukan.

Atribut-atribut yang diambil dari citra adalah rasio citra, keterbukaan citra, jumlah garis vertikal dan horizontal, garis perpotongan vertikal dan horizontal di bagian tengah citra, dan perbandingan *pixel* citra pada setiap blok dengan seluruh *pixel* citra.

3.2.5.4.2.1 Rasio Citra

Rasio citra dihitung dengan membandingkan lebar dan tinggi citra. Lebar didapatkan dari hasil selisih antara batas kanan dan kiri citra. Tinggi citra didapatkan dari selisih antara batas bawah dan atas citra.

Gambar 3.37 adalah *flowchart* algoritma yang digunakan untuk menghitung rasio citra.

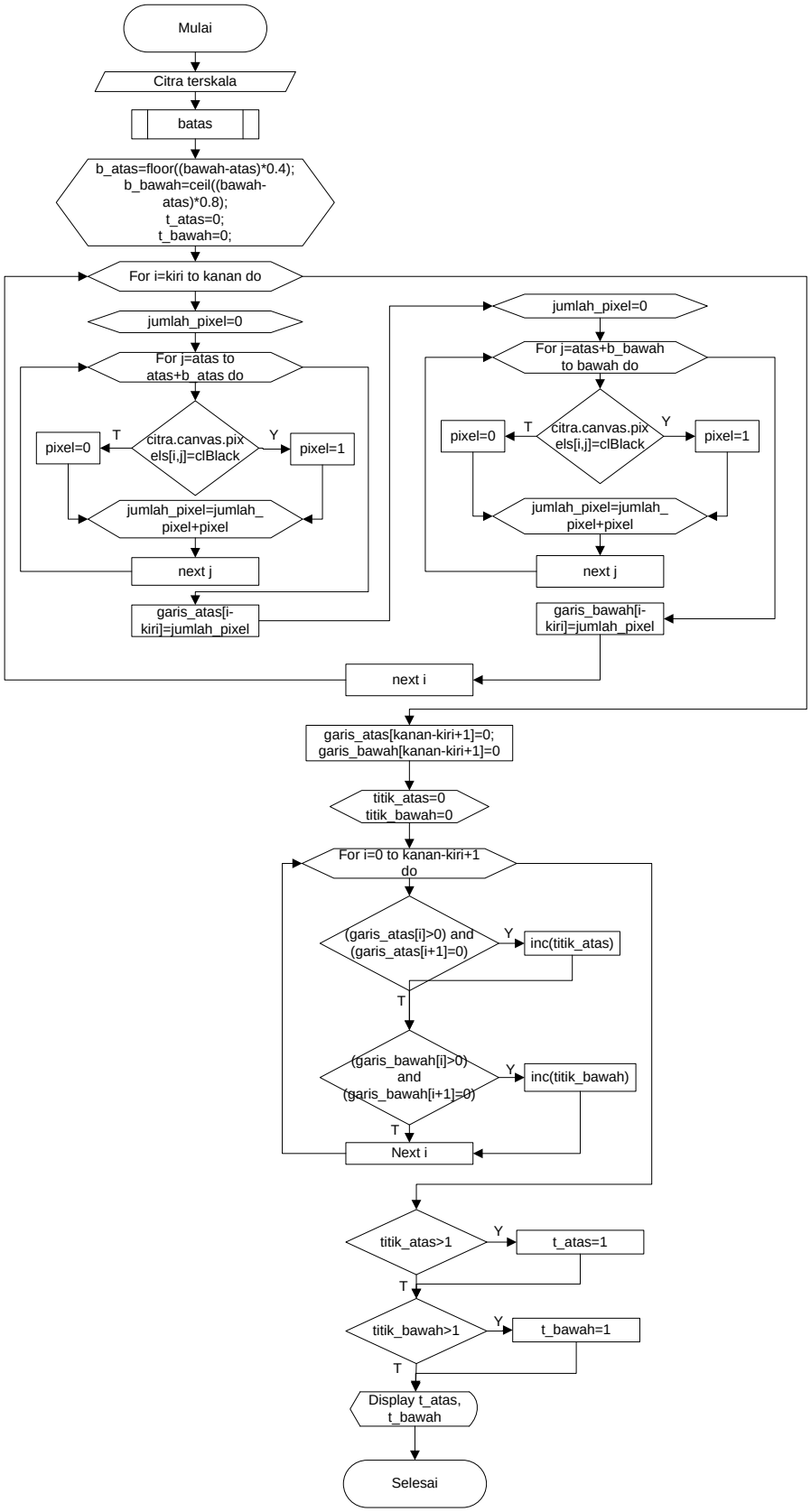


Gambar 3.37. *Flowchart* untuk Menghitung Rasio Citra.

Keterangan gambar 3.37:
1. ratio : rasio perbandingan antara lebar dan tinggi citra

3.2.5.4.2.2 Keterbukaan Citra

Gambar 3.38 adalah *flowchart* algoritma yang digunakan untuk mendeteksi apakah citra karakter terbuka ke atas atau ke bawah.



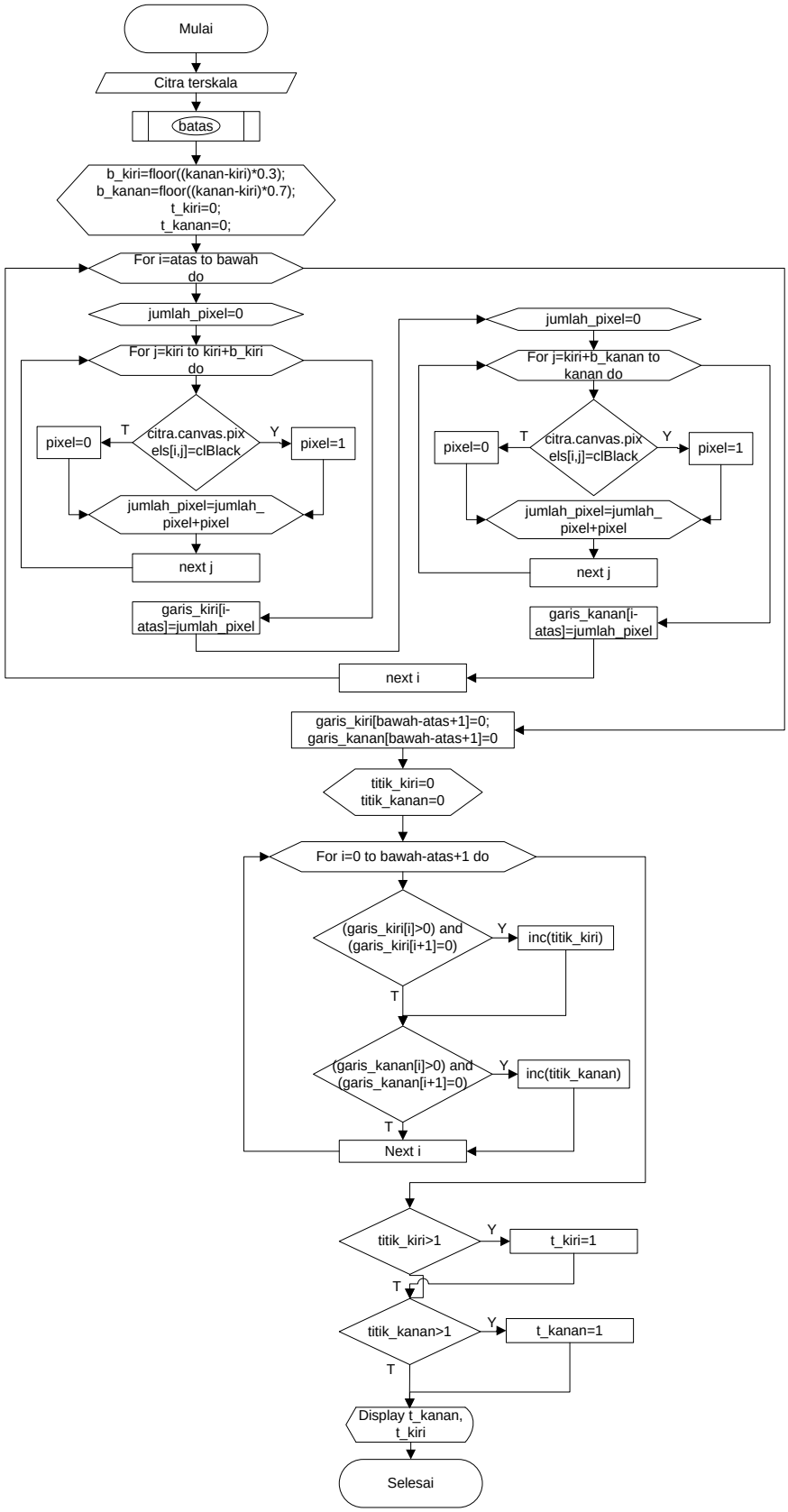
Gambar 3.38. *Flowchart* Algoritma Citra Terbuka Ke Atas atau Ke Bawah.

Keterangan gambar 3.38:

1. citra.canvas.pixel[j,i] : fungsi yang digunakan untuk mengakses intensitas pixel citra pada koordinat j,i.
2. garis_atas[i-kiri] : array yang digunakan untuk menyimpan jumlah pixel yang terdeteksi. [i-kiri] digunakan agar penyimpanan nilai perhitungan dimulai dari index ke-0.
3. garis_atas[kanan-kiri+1] : array index terakhir yang digunakan untuk menyimpan angka 0. Hal ini dibutuhkan saat mencari pola pixel.
4. garis_bawah[i-kiri] : array yang digunakan untuk menyimpan jumlah pixel yang terdeteksi. [i-kiri] digunakan agar penyimpanan nilai perhitungan dimulai dari index ke-0.
5. garis_bawah[kanan-kiri+1] : array index terakhir yang digunakan untuk menyimpan angka 0. Hal ini dibutuhkan saat mencari pola pixel.
6. titik_atas : variabel yang digunakan untuk menyimpan jumlah pola pixel 0 dan 1 di bagian atas citra (jika terdapat pola pixel 1 dan 0 sebanyak dua kali, maka citra dianggap terbuka).
7. titik_bawah : variabel yang digunakan untuk menyimpan jumlah pola pixel 0 dan 1 di bagian bawah citra (jika terdapat pola pixel 1 dan 0 sebanyak dua kali, maka citra dianggap terbuka).
8. t_atas : variabel yang digunakan untuk menyatakan bahwa citra terbuka ke atas (t_atas=1) atau tidak terbuka (t_atas=0).
9. t_bawah : variabel yang digunakan untuk menyatakan bahwa citra terbuka ke bawah (t_bawah=1) atau tidak terbuka (t_bawah=0).

Berdasarkan *flowchart* pada gambar 3.38 , untuk mendeteksi titik bagian atas, *scan* citra dengan cara menjumlahkan nilai pixel dari sisi terluar hingga mencapai 30% tinggi citra. Bagian bawah dideteksi dengan cara menelusuri pixel citra mulai dari 70% tinggi citra hingga mencapai 100% tinggi citra.

Gambar 3.39 adalah *flowchart* algoritma yang digunakan untuk mendeteksi apakah suatu citra karakter terbuka ke kiri atau ke kanan.



Gambar 3.39. *Flowchart* Algoritma Citra Terbuka Ke Kiri atau Ke Kanan.

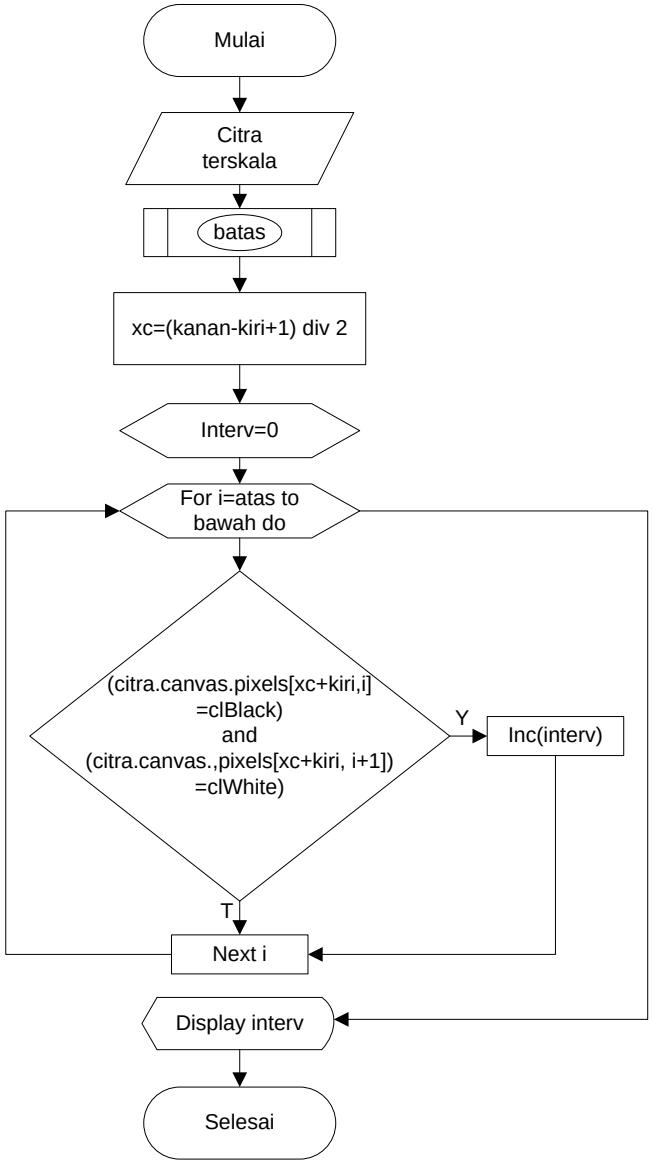
Keterangan gambar 3.39:

1. citra.canvas.pixel[j,i] : fungsi yang digunakan untuk mengakses intensitas pixel citra pada koordinat j,i.
2. garis_kiri[i-atas] : array yang digunakan untuk menyimpan jumlah pixel yang terdeteksi. [i-kiri] digunakan agar penyimpanan nilai perhitungan dimulai dari index ke-0.
3. garis_kiri[bawah-atas +1] : array index terakhir yang digunakan untuk menyimpan angka 0. Hal ini dibutuhkan saat mencari pola pixel.
4. garis_kanan[i-atas] : array yang digunakan untuk menyimpan jumlah pixel yang terdeteksi. [i-atas] digunakan agar penyimpanan nilai perhitungan dimulai dari index ke-0.
5. garis_kanan[bawah-atas +1] : array index terakhir yang digunakan untuk menyimpan angka 0. Hal ini dibutuhkan saat mencari pola pixel.
6. titik_kiri : variabel yang digunakan untuk menyimpan jumlah pola pixel 0 dan 1 di bagian kiri citra (jika terdapat pola pixel 1 dan 0 sebanyak dua kali, maka citra dianggap terbuka).
7. titik_kanan : variabel yang digunakan untuk menyimpan jumlah pola pixel 0 dan 1 di bagian kanan citra (jika terdapat pola pixel 1 dan 0 sebanyak dua kali, maka citra dianggap terbuka).
8. t_kiri : variabel yang digunakan untuk menyatakan bahwa citra terbuka ke kiri (t_kiri =1) atau tidak terbuka (t_kiri =0).
9. t_kanan : variabel yang digunakan untuk menyatakan bahwa citra terbuka ke kanan (t_kanan =1) atau tidak terbuka (t_kanan =0).

3.2.5.4.2.3 Jumlah Perpotongan Citra

Jumlah perpotongan citra terhadap garis vertikal dan horizontal di bagian tengah citra dihitung dengan cara menelusuri citra secara vertikal dan horizontal di bagian tengah citra. Jika suatu *pixel* berwarna hitam dan *pixel* selanjutnya berwarna putih, maka akan dihitung satu potongan. Hal ini dilakukan baik secara

vertikal dan horizontal. Gambar 3.40 adalah *flowchart* algoritma yang digunakan untuk mendeteksi jumlah perpotongan citra terhadap garis tengah secara vertikal.



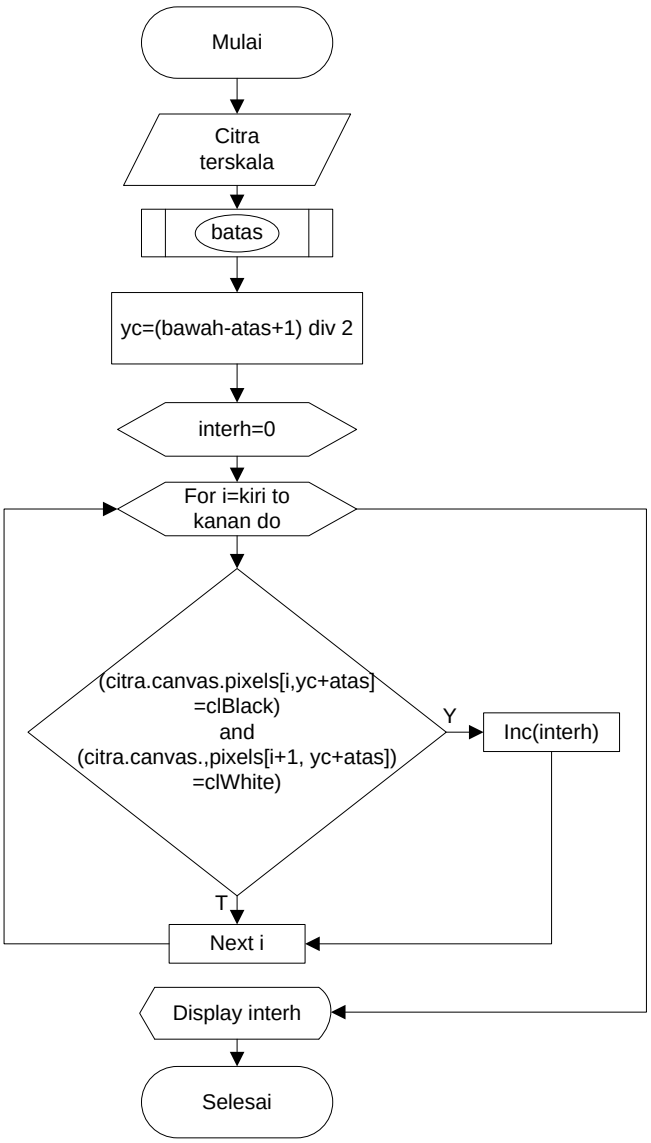
Gambar 3.40. *Flowchart* Algoritma Mendeteksi Perpotongan Garis Vertikal.

Keterangan gambar 3.40 :

1. xc : variabel yang digunakan untuk menyimpan koordinat vertikal bagian tengah citra.
2. interv : variabel yang digunakan untuk menyimpan jumlah perpotongan citra terhadap garis vertikal di bagian tengah.

3. citra.canvas.pixels[xc+kiri,i] : fungsi yang digunakan untuk mengakses pixel pada koordinat [xc+kiri,i]

Gambar 3.41 adalah *flowchart* algoritma yang digunakan untuk mendeteksi jumlah perpotongan citra terhadap garis tengah secara horizontal



Gambar 3.41. *Flowchart* Algoritma Mendeteksi Perpotongan Garis Horizontal.

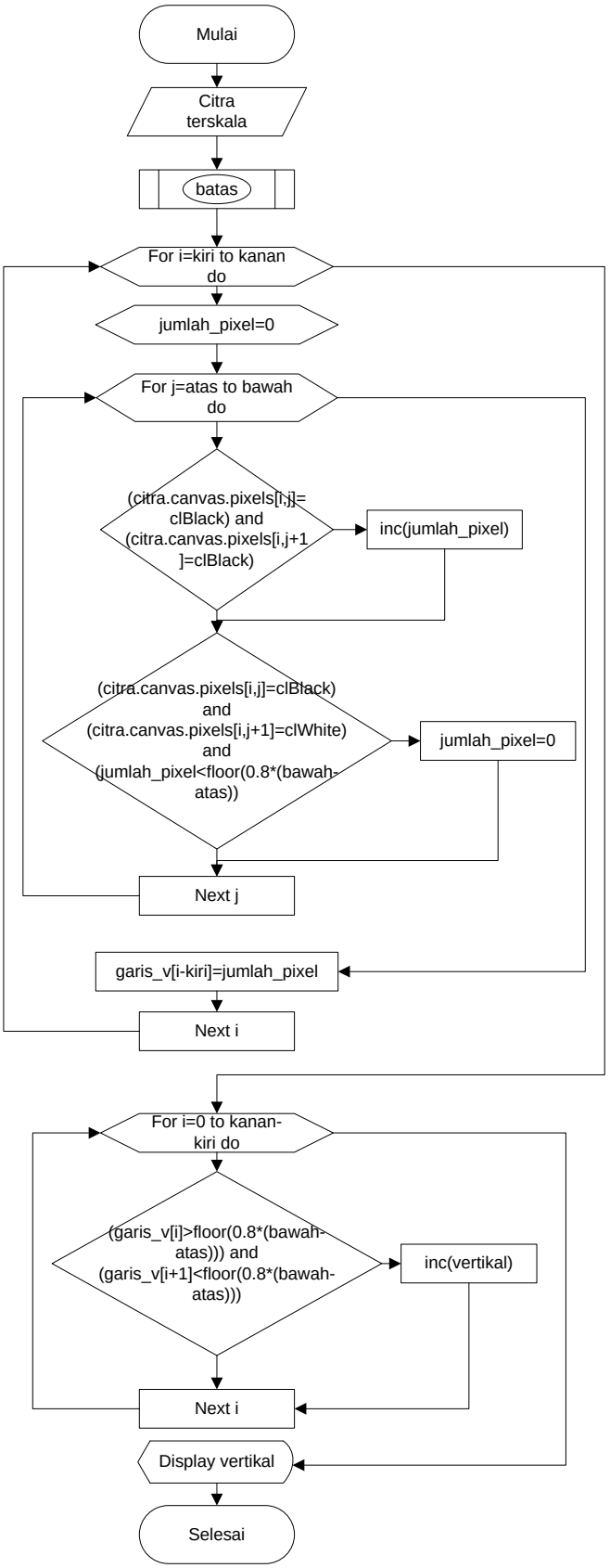
Keterangan gambar 3.41:

1. yc : variabel yang digunakan untuk menyimpan koordinat horizontal bagian tengah citra.

2. interh : variabel yang digunakan untuk menyimpan jumlah perpotongan citra terhadap garis horizontal di bagian tengah.
3. citra.canvas.pixels[yc+kiri,i] : fungsi yang digunakan untuk mengakses pixel pada koordinat [yc+kiri,i]

3.2.5.4.2.4 Menghitung Garis Vertikal dan Horizontal

Gambar 3.42 adalah *flowchart* algoritma yang digunakan untuk menghitung garis vertikal.



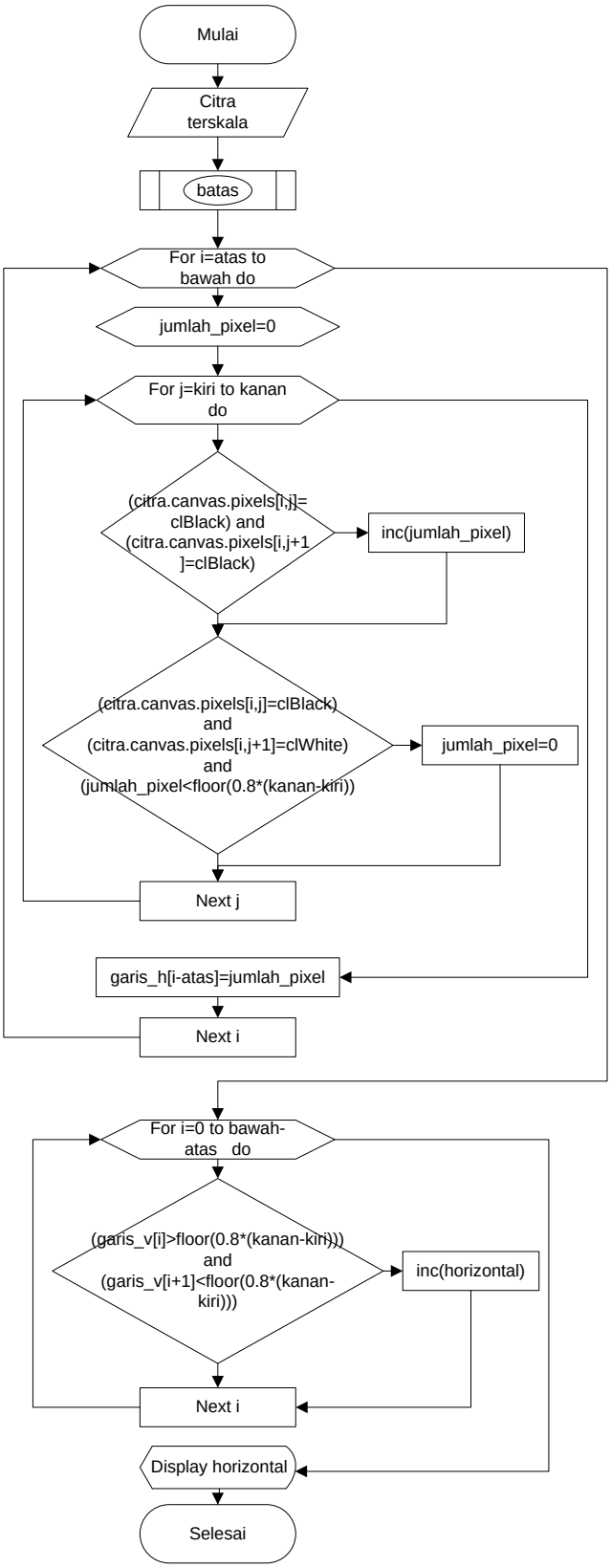
Gambar 3.42. *Flowchart* Algoritma untuk Mendeteksi Garis Vertikal.

Keterangan gambar 3.42:

1. garis_v[i-kiri] : array yang digunakakn untuk menyimpan jumlah pixel secara vertikal. [i-kiri] digunakan agar index array dimulai dari 0.
2. vertikal : variabel yang digunakan untuk menyimpan jumlah garis vertikal.
3. citra.canvas.pixels[i,j] : fungsi yang digunakan untuk mengakses pixel pada koordinat [i,j].

Berdasarkan *flowchart* algoritma pada gambar 3.42, garis vertikal dihitung dengan cara menjumlahkan nilai *pixel* secara vertikal. Jika jumlah nilai *pixel* lebih dari sama dengan 80% tinggi citra, maka citra tersebut memiliki garis vertikal. Jumlah pixel akan kembali menjadi 0 jika ditemukan pola pixel bernilai 0 sebelum jumlah pixel mencapai 80% tinggi citra karakter.

Gambar 3.43 adalah *flowchart* algoritma yang digunakan untuk menghitung garis horizontal suatu citra karakter.



Gambar 3.43. *Flowchart* Algoritma untuk Mendeteksi Garis Horizontal.

Keterangan gambar 3.43:

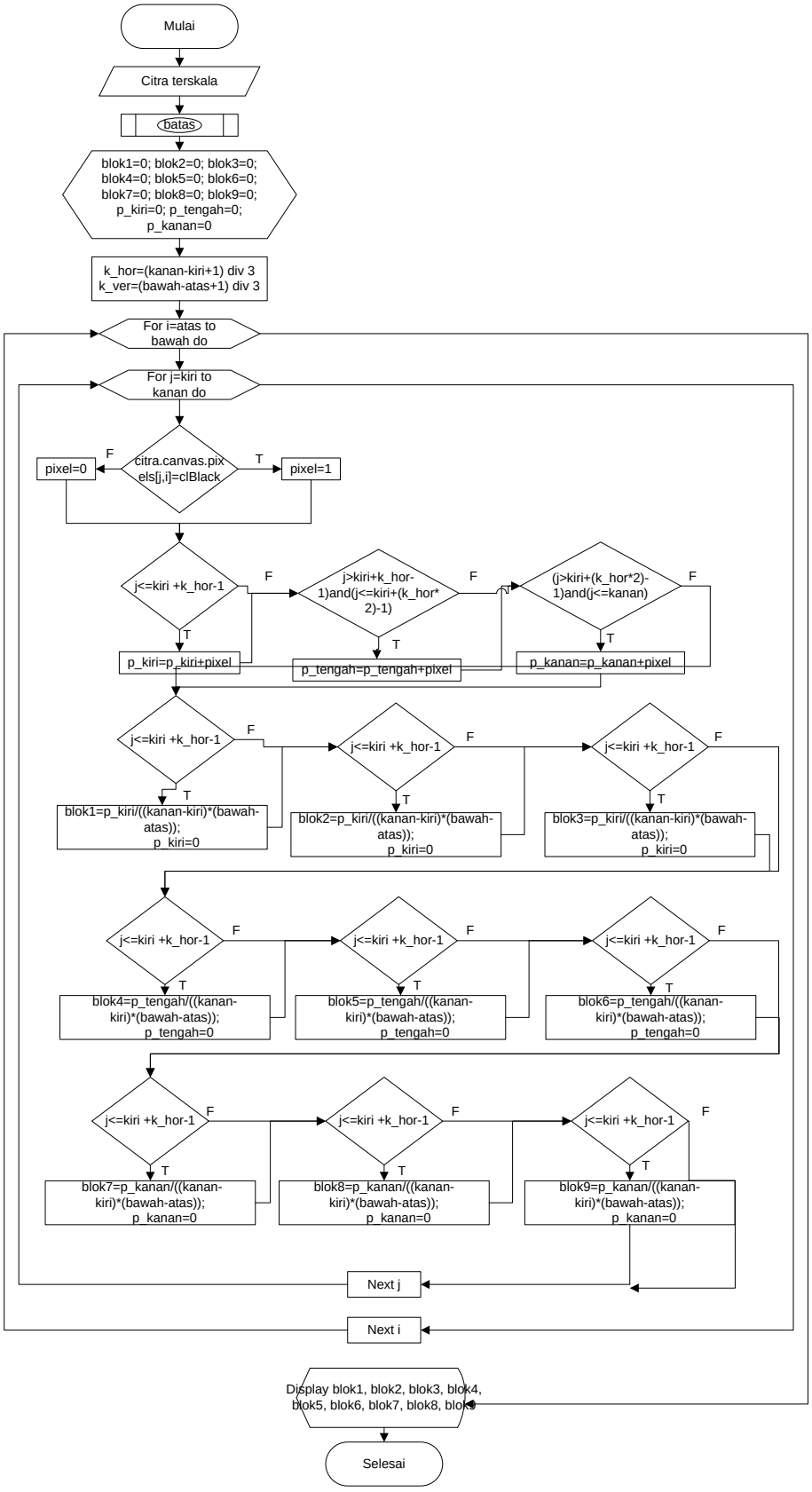
1. garis_h[i-atas] : array yang digunakakn untuk menyimpan jumlah pixel secara horizontal. [i-atas] digunakan agar index array dimulai dari 0.
2. horizontal : variabel yang digunakan untuk menyimpan jumlah garis horizontal.
3. citra.canvas.pixels[i,j] : fungsi yang digunakan untuk mengakses pixel pada koordinat [i,j].

Garis horizontal dihitung dengan cara menjumlahkan nilai pixel secara vertikal dan horizontal. Jika jumlah nilai pixel lebih dari sama dengan 80% lebar citra, maka citra tersebut memiliki garis horizontal. Jumlah pixel akan kembali menjadi 0 jika ditemukan pola pixel bernilai 0 sebelum jumlah pixel mencapai 80% lebar citra karakter.

3.2.5.4.2.5 Perbandingan *Pixel* pada Blok Citra

Citra dengan pixel berwarna hitam akan diwakili dengan angka 1 sedangkan citra yang berwarna putih akan diwakili dengan angka 0. Citra karakter dibagi menjadi sembilan bagian. Pada masing-masing bagian tersebut, dihitung jumlah nilai *pixel*nya dan dibandingkan dengan luas resolusi citra karakter.

Gambar 3.44 adalah *flowchart* algoritma yang digunakan untuk menghitung perbandingan jumlah nilai *pixel* suatu bagian dengan luas resolusi citra karakter.



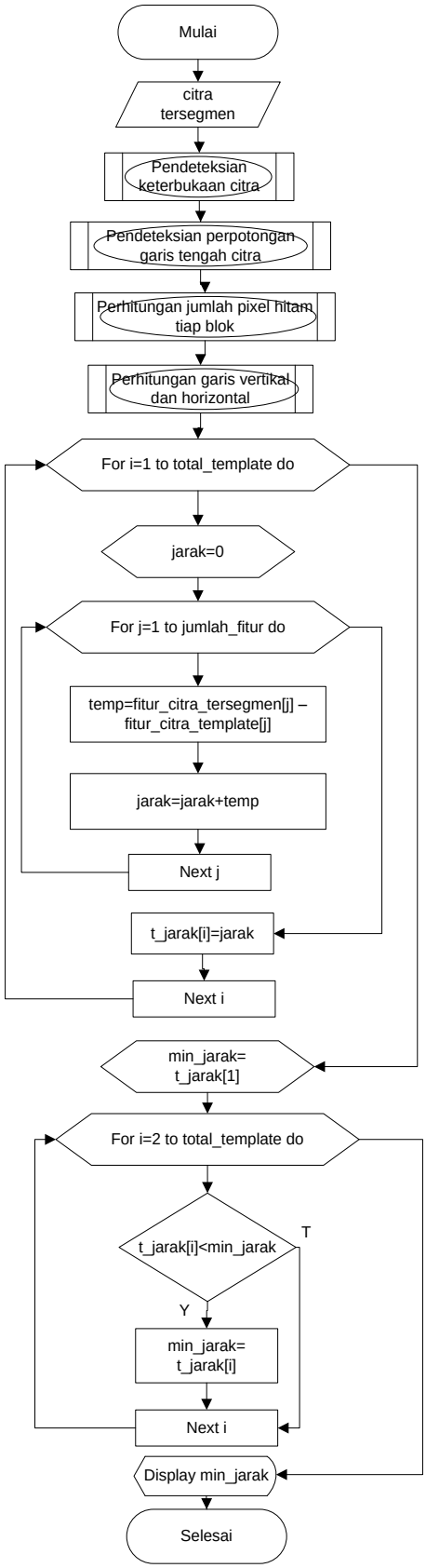
Gambar 3.44. *Flowchart* Algoritma untuk Menghitung Jumlah Pixel pada Blok Tertentu.

Keterangan gambar 3.44:

1. citra.canvas.pixels[i,j] : fungsi yang digunakan untuk mengakses pixel pada koordinat [i,j]
2. k_hor : variabel yang digunakan untuk menyimpan nilai pembatas vertikal.
3. k_ver : veriabal yang digunakan untuk menyimpan nilai pembatas horizontal.

3.2.5.4.2.6 Klasifikasi

Setelah citra karakter diambil ciri-cirinya, maka ciri-ciri tersebut d klasifikasikan dengan menggunakan metode k-NN. Metode k-NN menggunakan perhitungan jarak sesuai dengan persamaan 2-7. Jarak masing-masing fitur setiap *template* dihitung dengan masing-masing fitur citra karakter yang akan dikenali. Gambar 3.45 adalah flowchart algoritma klasifikasi dengan menggunakan persamaan 2-7.



Gambar 3.45. Flowchart Klasifikasi Fitur *Template* dan Fitur Citra Karakter yang akan Dikenali.

Keterangan gambar 3.45:

1. fitur_citra_tersegmen : fitur-fitur citra karakter yang akan dikenali.
2. fitur_citra_*template* : fitur-fitur citra karakter *template* yang ada pada basis data.
3. temp= hasil perhitungan jarak antara fitur citra yang akan dikenali dan fitur citra *template*.
4. t_jarak[i] : array yang digunakan untuk menyimpan nilai jarak masing-masing *template*.
5. min_jarak: variabel yang digunakan untuk menyimpan nilai jarak terkecil

BAB 4
IMPLEMENTASI DAN PENGUJIAN SISTEM

Tahap implementasi dilaksanakan setelah tahap perancangan sistem. Tahap implementasi merupakan tahap meletakkan sistem upaya siap untuk dioperasikan dan dapat dipandang sebagai usaha untuk mewujudkan sistem yang telah dirancang.

4.1 Implementasi Sistem

Tahapan-tahapan implementasi berupa implementasi perangkat keras, implementasi perangkat lunak, implementasi basis data, implementasi antar muka, dan pengujian sistem.

4.1.1 Implementasi Perangkat Keras

Spesifikasi perangkat keras (*hardware*) yang digunakan untuk pembuatan aplikasi perbandingan metode ini meliputi :

1. Prosessor AMD 1.6 Ghz.
2. Media penyimpanan (*Harddisk*) 20 GB.
3. Memori *RAM* 256 Mb.
4. Kartu grafis 32 Mb.
5. Monitor 15 inchi.

4.1.2 Implementasi Perangkat Lunak

Spesifikasi perangkat lunak (*software*) yang digunakan untuk membuat aplikasi pengidentifikasian kata ini adalah :

1. Sistem operasi Window XP SP2.
2. Borland Delphi 7.
3. Wamp Server 2.0.
4. SQLyog Enterprise 8.05.
5. ODBC 5.1.

4.1.3 Implementasi Basis Data

Berikut ini adalah implementasi basis data yang digunakan dalam membangun aplikasi perbandingan metode:

1. Tabel t_eror

```
CREATE TABEL `t_eror` (
  `id_char` varchar(8) DEFAULT NULL,
  `karakter` char(1) DEFAULT NULL,
  `pixel` varchar(150) DEFAULT NULL,
  `eror` int(11) DEFAULT NULL,
  `id_font` varchar(5) DEFAULT NULL,
  KEY `FK_t_char` (`id_font`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

2. Tabel t_*feature*_geometri

```
CREATE TABEL `t_feature_geometri` (
  `id` int(11) DEFAULT NULL,
  `kar` varchar(3) DEFAULT NULL,
  `ratio` float DEFAULT NULL,
  `t_kiri` int(11) DEFAULT NULL,
  `t_kanan` int(11) DEFAULT NULL,
  `t_atas` int(11) DEFAULT NULL,
  `t_bawah` int(11) DEFAULT NULL,
  `int_v` int(11) DEFAULT NULL,
  `int_h` int(11) DEFAULT NULL,
  `garis_v` int(11) DEFAULT NULL,
  `garis_h` int(11) DEFAULT NULL,
  `pixel_atas` float DEFAULT '0',
```

```
        `pixel_kanan` float DEFAULT '0',
        `blok1` float DEFAULT NULL,
        `blok2` float DEFAULT NULL,
        `blok3` float DEFAULT NULL,
        `blok4` float DEFAULT NULL,
        `blok5` float DEFAULT NULL,
        `blok6` float DEFAULT NULL,
        `blok7` float DEFAULT NULL,
        `blok8` float DEFAULT NULL,
        `blok9` float DEFAULT NULL,
        `jarak` float DEFAULT '0',
        `font`   char(1)   DEFAULT   'A')   ENGINE=InnoDB   DEFAULT
CHARSET=latin1;
```

3. Tabel t_*font*

```
      CREATE TABEL `t_font` (
        `id_font` varchar(5) NOT NULL,
        `nama_font` varchar(50) DEFAULT NULL,
        `ukuran` int(11) DEFAULT NULL,
        PRIMARY KEY (`id_font`)
      ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

4.1.4 Implementasi Antar Muka

Berikut ini adalah daftar tampilan form aplikasi Perbandingan Metode *Template Matching* dan *Feature Extraction* pada *Optical Character Recognition*.

Tabel 4.1. Daftar Form.

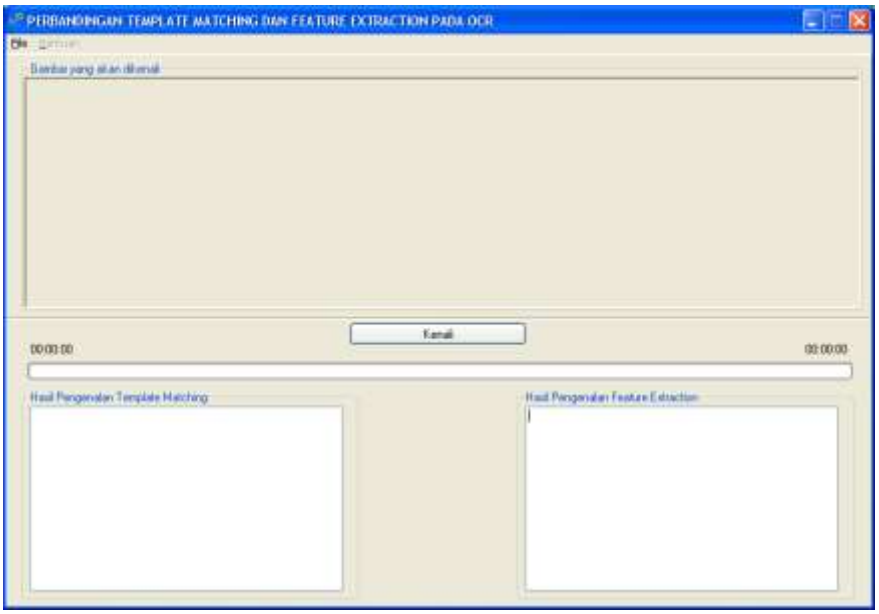
No	Nama Form	Keterangan
1	Form Utama	Form utama untuk proses pengenalan.
2	Cara Penggunaan	Form cara menggunakan aplikasi
3	Tentang aplikasi	Form yang berisi informasi mengenai aplikasi yang dibangun.

Berikut adalah tampilan antar muka pengguna pada aplikasi perbandingan metode:

1. Form utama

   Form ini digunakan untuk menampilkan citra yang akan dikenali dan hasil pengenalan kedua metode. Form ini dilengkapi dengan menu,

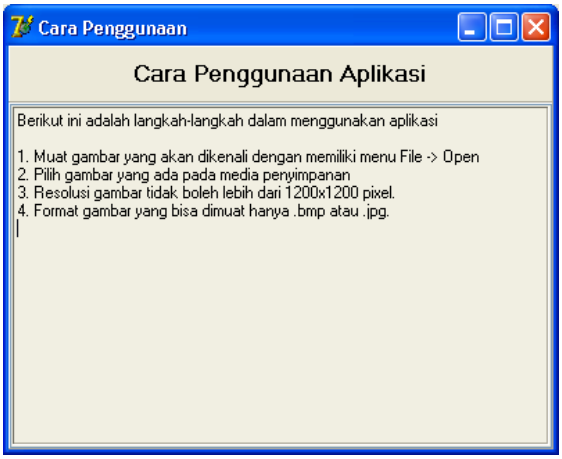
tombol, progress bar, jendela untuk menampilkan gambar, dan jendela untuk menampilkan hasil pengenalan kedua metode.



Gambar 4.1. Tampilan Form Utama.

2. Form cara penggunaan

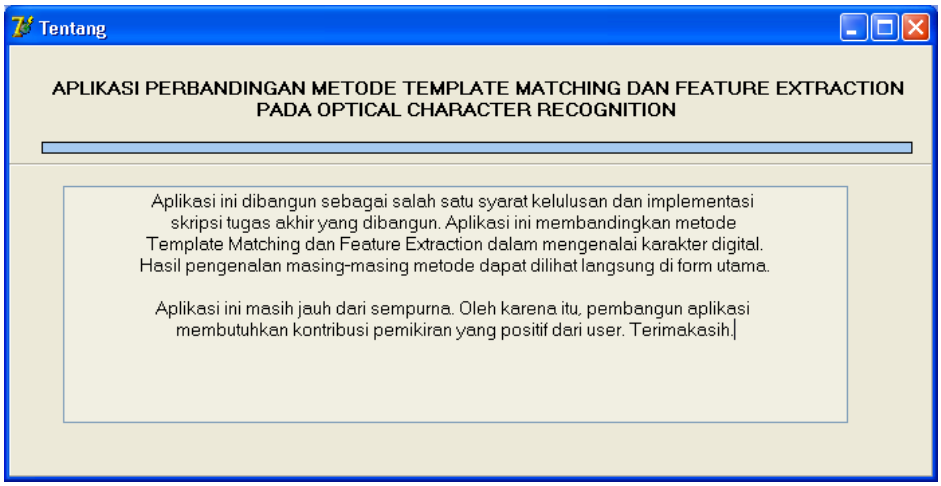
   Berikut ini adalah form yang digunakan untuk menginformasikan user cara menggunakan program.



Gambar 4.2. Tampilan Form Cara Penggunaan.

3. Form Tentang

Form ini digunakan untuk mendeskripsikan aplikasi yang dibangun secara singkat.



Gambar 4.3. Tampilan Form Tentang.

4.2 Pengujian Sistem

Setelah melakukan tahapan implementasi, maka tahapan selanjutnya yaitu pengujian program yang telah dibuat. Sistem akan diuji dengan pengujian *alpha*.

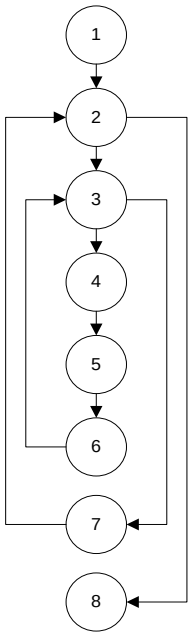
Kedua algoritma akan diuji dengan menggunakan dokumen uji yang memiliki empat *font* berbeda dengan tiga *font size* dan tulisan tangan. Pengukuran hasil pengujian adalah persentase keakuratan dan waktu yang dibutuhkan masing-masing algoritma terhadap dokumen uji.

4.2.1 Kasus dan Hasil Pengujian *Alpha*

Pengujian alpha dilakukan dengan menggunakan uji coba *basis path* dengan menggunakan diagram alir dan menghitung kompleksitas siklomatis pada masing-masing proses. Kompleksitas siklomatis dihitung dengan menggunakan simpul predikat. Sehingga jumlah diagram alir harus sama dengan simpul predikat yang ditambah dengan 1.

4.2.1.1 Pengujian Proses *Grayscaling*

Gambar 4.4 adalah notasi diagram alir pada proses *grayscale*.



Gambar 4.4. Diagram Alir Proses *Grayscaling*.

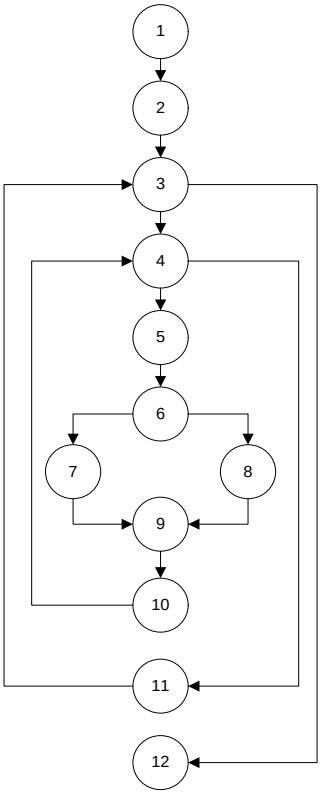
Tabel 4.2 adalah hasil jalur independen dan perhitungan kompleksitas siklomatis berdasarkan gambar 4.4.

Tabel 4.2. Perhitungan Jalur Independen dan Kompleksitas Siklomatis.

Jalur independen		Perhitungan siklomatis		Jumlah jalur grafik=siklomatis
No.	Jalur	Simpul predikat (P)	V(G)=P+1	
1	1-2-3-4-5-6-3-7-2-8	2	3	[√] diterima [ ] ditolak
2	1-2-8			
3	1-2-3-7-2-8			

4.2.1.2 Pengujian Proses Binerisasi

Gambar 4.5 adalah notasi diagram alir pada proses binerisasi.



Gambar 4.5. Diagram Alir Proses Binerisasi.

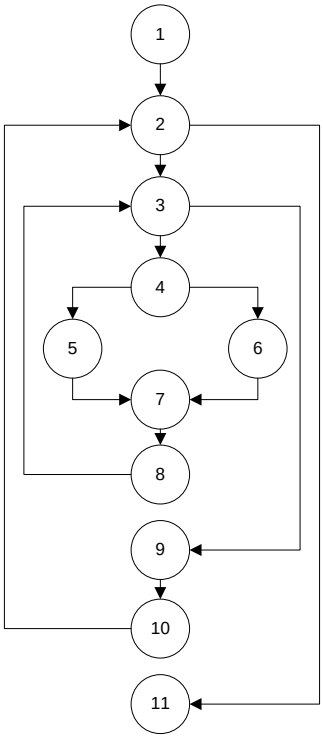
Tabel 4.3 adalah hasil jalur independen dan perhitungan siklomatis berdasarkan gambar 4.5.

Tabel 4.3. Perhitungan jalur independen dan kompleksitas siklomatis.

Jalur independen		Perhitungan siklomatis		Jumlah jalur grafik=siklomatis
No.	Jalur	Simpul predikat (P)	V(G)=P+1	
1	1-2-3-4-5-6-7-9-10-4-11-3-12	3	4	[√] diterima [ ] ditolak
2	1-2-3-12			
3	1-2-3-4-11-3-12			
4	1-2-3-4-5-6-8-9-10-4-11-3-12			

4.2.1.3 Pengujian Proses Segmentasi Baris

Gambar 4.6 adalah notasi diagram alir pada proses segmentasi baris.



Gambar 4.6. Diagram Alir Proses Segmentasi Baris.

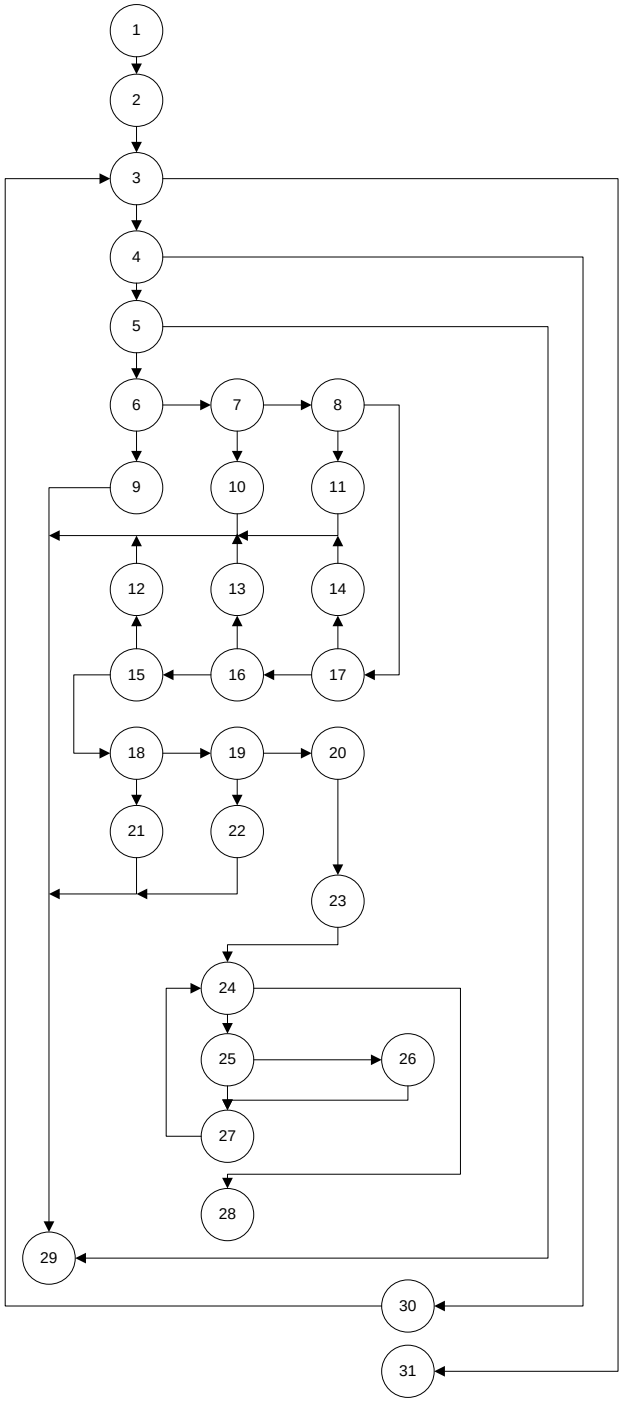
Tabel 4.4 adalah hasil jalur independen dan perhitungan siklomatis berdasarkan gambar 4.6.

Tabel 4.4. Perhitungan Jalur Independen dan Kompleksitas Siklomatis.

Jalur independen		Perhitungan siklomatis		Jumlah jalur grafik=siklomatis
No.	Jalur	Simpul predikat (P)	V(G)= P+1	
1	1-2-11	3	4	[√] diterima [ ] ditolak
2	1-2-3-9-10-2-11			
3	1-2-3-4-5-7-8-3-9-10-2-11			
4	1-2-3-4-6-7-8-3-9-10-2-11			

4.2.1.4 Pengujian Proses Segmentasi Karakter

Gambar 4.7 adalah notasi diagram alir segmentasi karakter yang mengguakan metode *connected component analysis*.



Gambar 4.7. Diagram Alir Proses *Connected Component Analysis*.

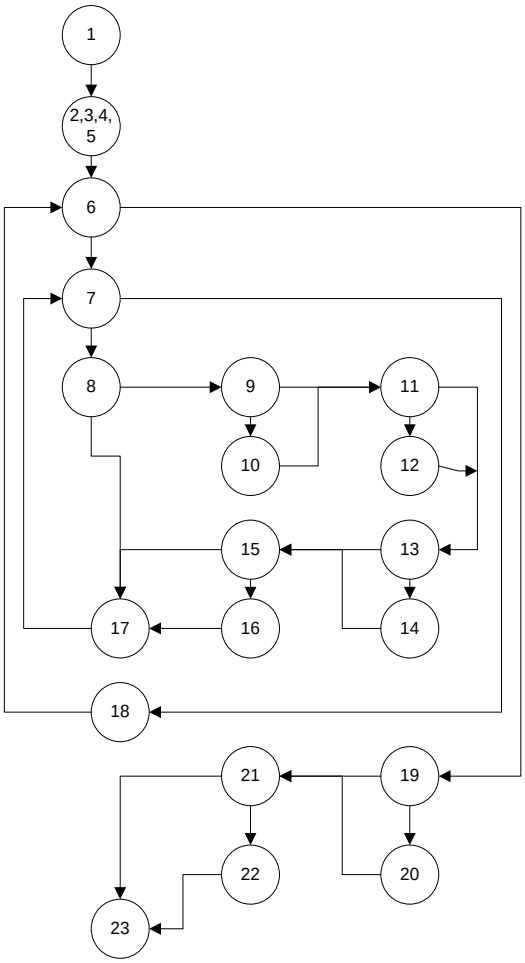
Tabel 4.5 adalah hasil jalur independen dan perhitungan siklomatis berdasarkan gambar 4.7.

Tabel 4.5. Perhitungan Jalur Independen dan Kompleksitas Siklomatis.

Jalur independen		Perhitungan siklomatis		Jumlah jalur grafik=siklomatis
No.	Jalur	Simpul Predikat (P)	V(G)= P+1	
1.	1-2-3-31	14	15	[√] diterima [ ] ditolak
2.	1-2-3-4-30-3-31			
3.	1-2-3-4-5-29-4-30-3-31			
4.	1-2-3-4-5-6-7-8-17-16-15-18-19-20-23-24-25-27-28-23-29-4-30-3-31			
5.	1-2-3-4-5-6-7-8-17-16-15-18-19-20-23-24-25-27-28-23-29-4-30-3-31			
6.	1-2-3-4-5-6-9-29-4-30-3-31			
7.	1-2-3-4-5-6-7-10-29-4-30-3-31			
8.	1-2-3-4-5-6-7-8-11-29-4-30-3-31			
9.	1-2-3-4-5-6-7-8-17-14-29-4-30-3-31			
10.	1-2-3-4-5-6-7-8-17-16-13-29-4-30-3-31			
11.	1-2-3-4-5-6-7-8-17-16-15-12-29-4-30-3-31			
12.	1-2-3-4-5-6-7-8-17-16-15-18-21-29-4-30-3-31			
13.	1-2-3-4-5-6-7-8-17-16-15-18-19-22-29-4-30-3-31			
14.	1-2-3-4-5-6-7-8-17-16-15-18-19-20-23-29-4-30-3-31			
15.	1-2-3-4-5-6-7-8-17-16-15-18-19-20-23-24-25-26-27-24-28-23-29-4-30-3-31			

4.2.1.5 Pengujian Proses Pencarian Batas

Gambar 4.8 adalah notasi diagram alir pada proses pencarian batas-batas citra karakter.



Gambar 4.8. Diagram Alir Proses Pencarian Batas.

Tabel 4.6 adalah hasil jalur independen dan perhitungan siklomatis berdasarkan gambar 4.8.

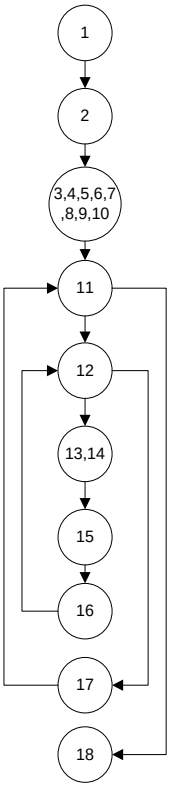
Tabel 4.6. Perhitungan jalur independen dan kompleksitas siklomatis.

Jalur independen		Perhitungan siklomatis		Jumlah jalur grafik=siklomatis
No.	Jalur	Simpul Predikat (P)	V(G)=P+1	
1	1-2,3,4,5-6-19-21-23	9	10	[√] diterima [ ] ditolak
2	1-2.3.4.5-6-7-18-6-19-21-23			
3	1-2,3,4,5-6-7-8-17-7-18-6-19-21-23			
4	1-2,3,4,5-6-7-8-9-11-13-15-17-7-18-6-19-21-23			
5	1-2,3,4,5-6-7-8-9-10-11-13-15-17-7-18-6-19-21-23			
6	1-2,3,4,5-6-7-8-9-11-12-13-15-17-7-18-6-19-21-23			

Jalur independen		Perhitungan siklomatis		Jumlah jalur grafik=siklomatis
No.	Jalur	Simpul Predikat (P)	V(G)=P+1	
7	1-2,3,4,5-6-7-8-9-11-13-14-15-17-7-18-6-19-21-23			
8	1-2,3,4,5-6-7-8-9-11-13-15-16-17-7-18-6-19-21-23			
9	1-2,3,4,5-6-7-8-9-11-13-15-17-7-18-6-19-20-21-23			
10	1-2,3,4,5-6-7-8-9-11-13-15-17-7-18-6-19-21-22-23			

4.2.1.6 Pengujian Proses Penskalaan

Gambar 4.9 adalah notasi diagram alir pada proses penskalaan.



Gambar 4.9. Diagram Alir Proses Penskalaan.

Tabel 4.7 adalah hasil jalur independen dan perhitungan siklomatis berdasarkan gambar 4.9.

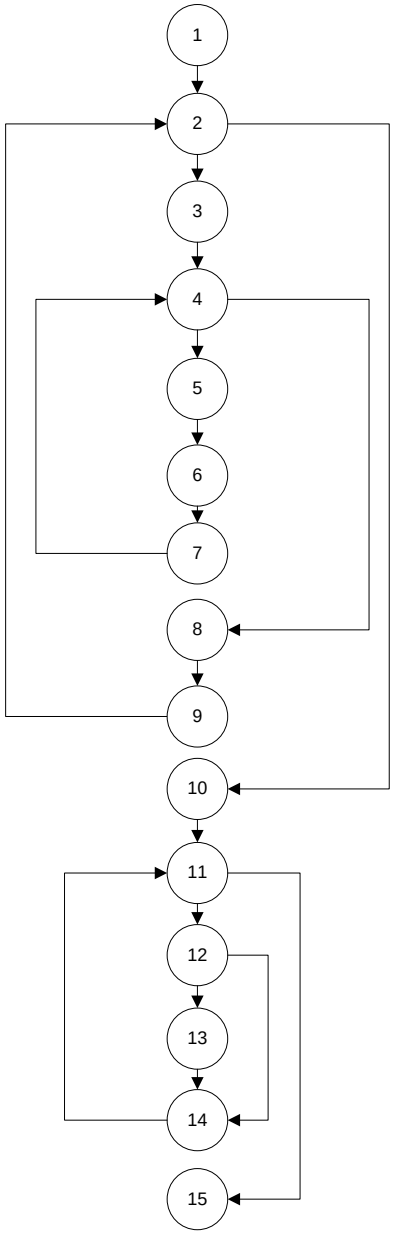
Tabel 4.7. Perhitungan Jalur Independen dan Kompleksitas Siklomatis.

Jalur independen		Perhitungan siklomatis		Jumlah jalur grafik=siklomatis
No.	Jalur	Simpul	V(G)=P+1	

		Predikat (P)		
1	1-2-3,4,5,6,7,8,9,10-11-12-13,14-15-16-12-17-11-18	2	3	[√] diterima [ ] ditolak
2	1-2-3,4,5,6,7,8,9,10-11-18			
3	1-2-3,4,5,6,7,8,9,10-11-12-17-11-18			

4.2.1.7 Pengujian Proses Pengenalan dengan *Template matching*

Gambar 4.10 adalah notasi diagram alir pada proses *template matching*.



Gambar 4.10. Diagram Alir Proses *Template Matching*.

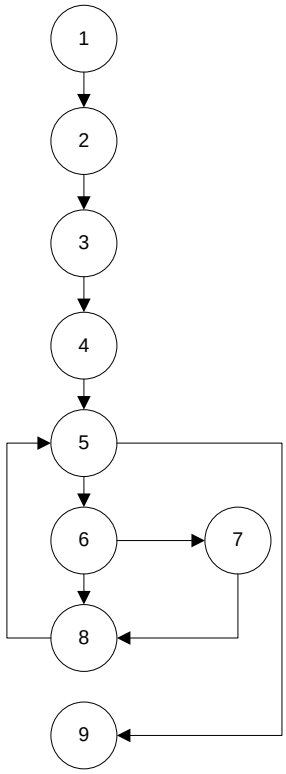
Tabel 4.8 adalah hasil jalur independen dan perhitungan siklomatis berdasarkan gambar 4.10.

Tabel 4.8. Perhitungan Jalur Independen dan Kompleksitas Siklomatis.

Jalur independen		Perhitungan siklomatis		Jumlah jalur grafik=siklomatis
No.	Jalur	Simpul Predikat (P)	V(G)=P+1	
1	1-2-3-4-5-6-7-4-8-9-2-10-11-12-13-14-11-15	4	5	[√] diterima [ ] ditolak
2	1-2-10-11-12-13-14-11-15			
3	1-2-3-4-5-6-7-4-8-9-2-10-11-15			
4	1-2-3-4-8-9-2-10-11-12-13-14-11-15			
5	1-2-3-4-5-6-7-4-8-9-2-10-11-12-14-11-15			

4.2.1.8 Pengujian Proses Pendeteksian Perpotongan Garis Tengah citra

Gambar 4.11 adalah notasi diagram alir pada proses pendeteksian perpotongan garis tengah citra secara vertikal dan horizontal.



Gambar 4.11. Diagram Alir Proses Pendeteksian Perpotongan Garis Tengah Citra Secara Vertikal dan Horizontal.

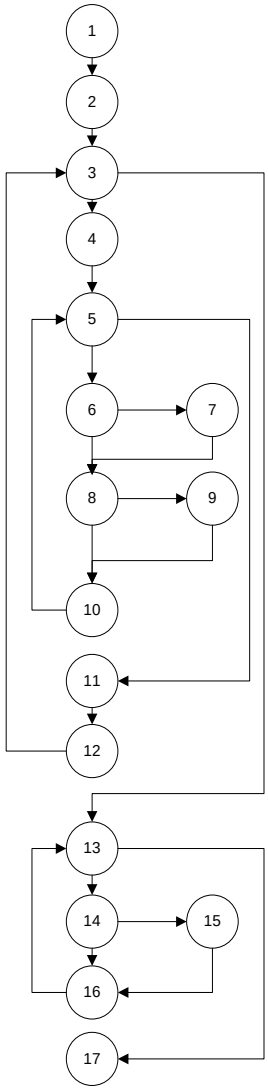
Tabel 4.9 adalah hasil jalur independen dan perhitungan siklomatis berdasarkan gambar 4.11.

Tabel 4.9. Perhitungan Jalur Independen dan Kompleksitas Siklomatis.

Jalur independen		Perhitungan siklomatis		Jumlah jalur grafik=siklomatis
No.	Jalur	Simpul Predikat (P)	V(G)=P+1	
1	1-2-3-4-5-6-8-5-9	2	3	[√] diterima [ ] ditolak
2	1-2-3-4-5-9			
3	1-2-3-4-5-6-7-8-5-9			

4.2.1.9 Pengujian Proses Perhitungan Jumlah Garis Vertikal dan Horizontal

Gambar 4.12 adalah notasi diagram alir pada proses perhitungan jumlah garis vertikal dan horizontal.



Gambar 4.12. Diagram Alir Proses Perhitungan Jumlah Garis Vertikal dan Horizontal.

Tabel 4.10 adalah hasil jalur independen dan perhitungan siklomatis berdasarkan gambar 4.12.

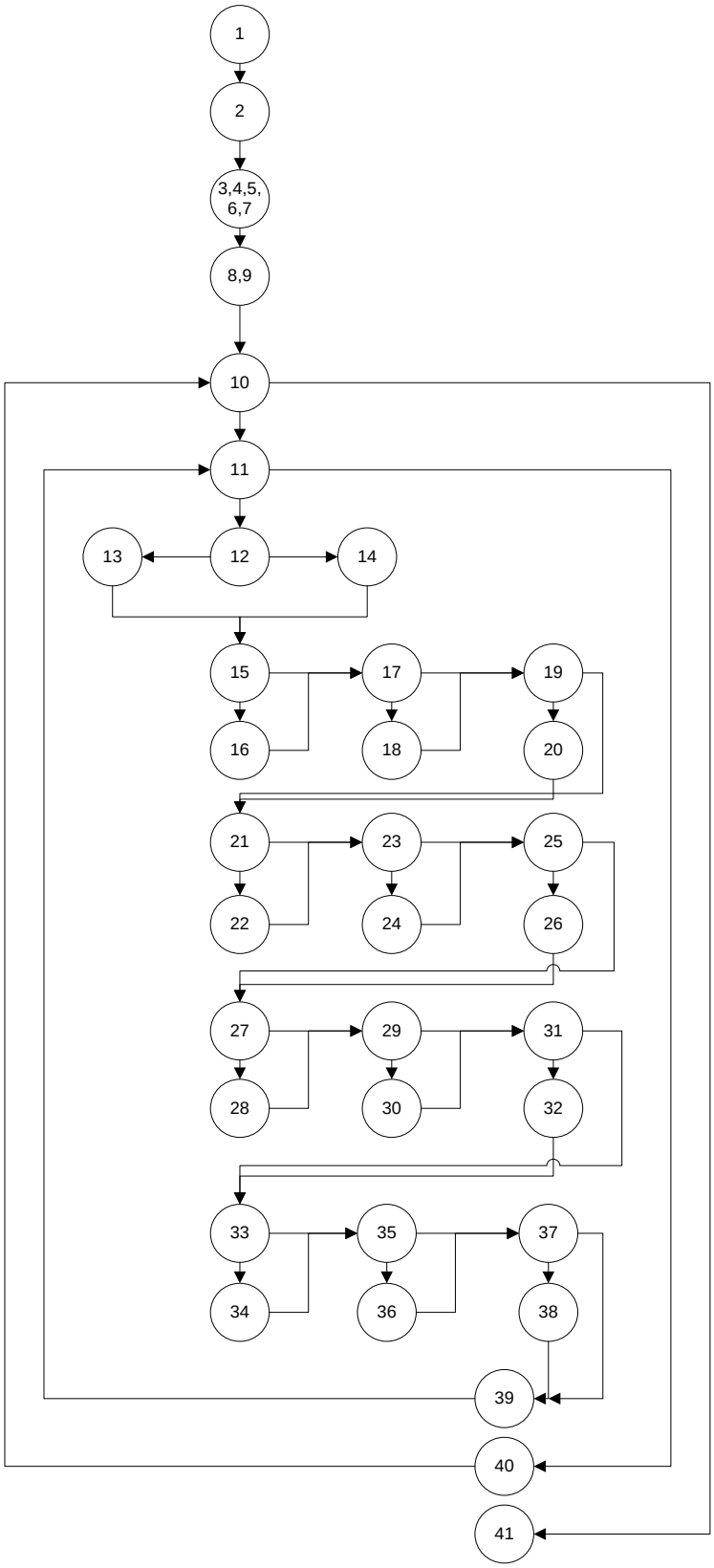
Tabel 4.10. Perhitungan Jalur Independen dan Kompleksitas Siklomatis.

Jalur independen		Perhitungan siklomatis		Jumlah jalur grafik=siklomatis
No.	Jalur	Simpul Predikat (P)	V(G)=P+1	
1	1-2-3-4-5-6-8-10-5-11-12-3-13-14-16-13-17	6	7	[√] diterima [ ] ditolak
2	1-2-3-4-5-11-12-3-13-14-16-13-17			

3	1-2-3-13-14-16-13-17			
4	1-2-3-4-5-6-8-10-5-11-12-3-13-17			
5	1-2-3-4-5-6-7-8-10-5-11-12-3-13-14-16-13-17			
6	1-2-3-4-5-6-8-9-10-5-11-12-3-13-14-16-13-17			
7	1-2-3-4-5-6-8-10-5-11-12-3-13-14-15-16-13-17			

4.2.1.10 Pengujian Proses Perhitungan Jumlah Pixel Hitam Tiap Blok

Gambar 4.13 adalah notasi diagram alir pada proses perhitungan jumlah pixel hitam tiap blok.



Gambar 4.13. Diagram Alir Proses Perhitungan Jumlah Pixel Hitam Tiap Blok.

Tabel 4.11 adalah hasil jalur independen dan perhitungan siklomatis berdasarkan gambar 4.13.

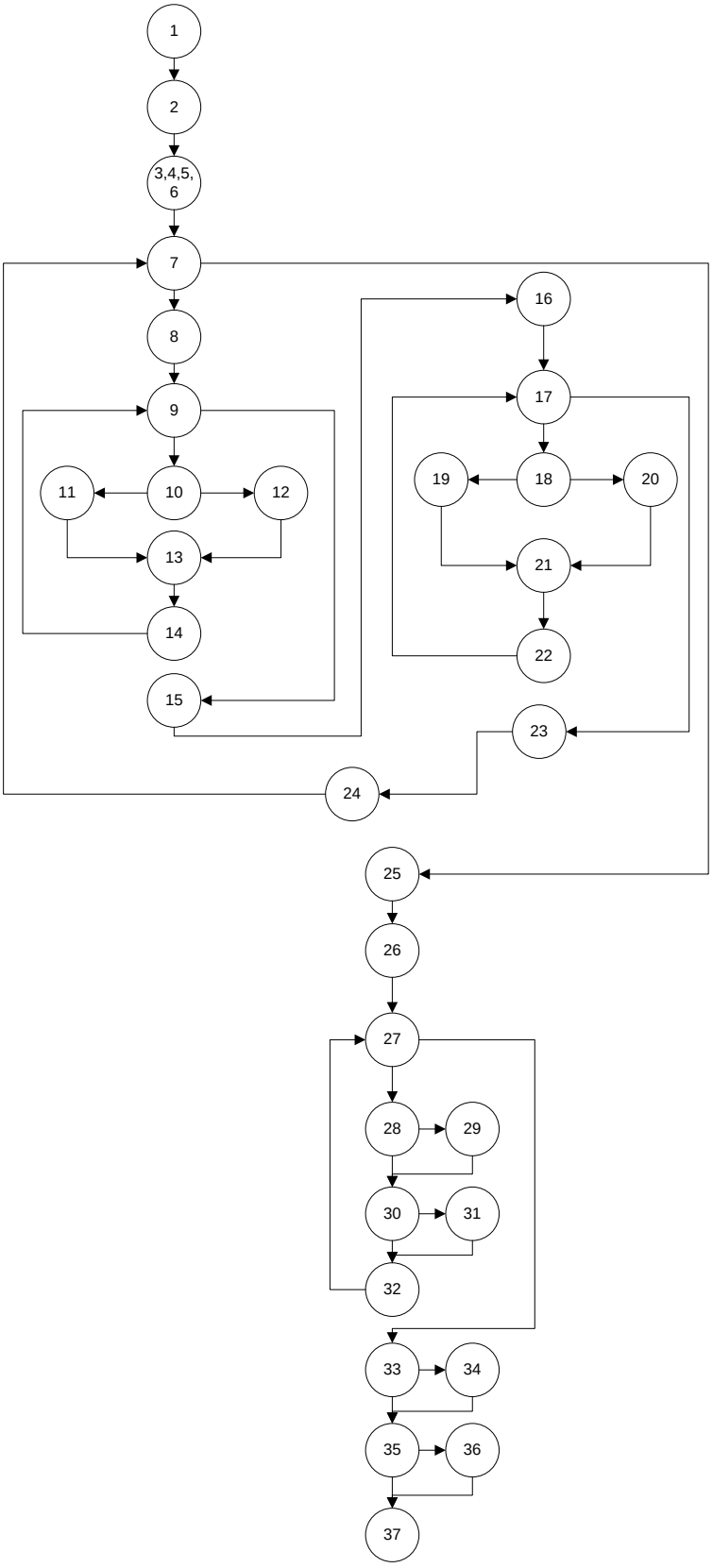
Tabel 4.11. Perhitungan Jalur Independen dan Kompleksitas Siklomatis.

Jalur independen		Perhitungan siklomatis		Jumlah jalur grafik=siklomatis
No.	Jalur	Simpul Predikat (P)	V(G)=P+1	
1	1-2-3,4,5,6,7-8,9-10-41	15	16	[√] diterima [ ] ditolak
2	1-2-3,4,5,6,7-8,9-11-40-10-41			
3	1-2-3,4,5,6,7-8,9-11-12-14-15-17-19-21-23-25-27-29-31-33-35-37-39-11-40-10-41			
4	1-2-3,4,5,6,7-8,9-11-12-13-15-17-19-21-23-25-27-29-31-33-35-37-39-11-40-10-41			
5	1-2-3,4,5,6,7-8,9-11-12-14-15-16-17-19-21-23-25-27-29-31-33-35-37-39-11-40-10-41			
6	1-2-3,4,5,6,7-8,9-11-12-14-15-17-18-19-21-23-25-27-29-31-33-35-37-39-11-40-10-41			
7	1-2-3,4,5,6,7-8,9-11-12-14-15-17-19-20-21-23-25-27-29-31-33-35-37-39-11-40-10-41			
8	1-2-3,4,5,6,7-8,9-11-12-14-15-17-19-21-22-23-25-27-29-31-33-35-37-39-11-40-10-41			
9	1-2-3,4,5,6,7-8,9-11-12-14-15-17-19-21-23-24-25-27-29-31-33-35-37-39-11-40-10-41			
10	1-2-3,4,5,6,7-8,9-11-12-14-15-17-19-21-23-25-26-27-29-31-			

Jalur independen		Perhitungan siklomatis		Jumlah jalur grafik=siklomatis
No.	Jalur	Simpul Predikat (P)	V(G)=P+1	
	33-35-37-39-11-40-10-41			
11	1-2-3,4,5,6,7-8,9-11-12-14-15-17-19-21-23-25-27-28-29-31-33-35-37-39-11-40-10-41			
12	1-2-3,4,5,6,7-8,9-11-12-14-15-17-19-21-23-25-27-29-30-31-33-35-37-39-11-40-10-41			
13	1-2-3,4,5,6,7-8,9-11-12-14-15-17-19-21-23-25-27-29-31-32-33-35-37-39-11-40-10-41			
14	1-2-3,4,5,6,7-8,9-11-12-14-15-17-19-21-23-25-27-29--31-33-34-35-37-39-11-40-10-41			
15	1-2-3,4,5,6,7-8,9-11-12-14-15-17-19-21-23-25-27-29-31-33-35-36-37-39-11-40-10-41			
16	1-2-3,4,5,6,7-8,9-11-12-14-15-17-19-21-23-25-27-29-30-31-33-35-37-38-39-11-40-10-41			

4.2.1.11 Pengujian Proses Pendeteksian Keterbukaan Citra

Gambar 4.14 adalah notasi diagram alir pada proses pendeteksian keterbukaan citra.



Gambar 4.14. Diagram Alir Proses Pendeteksian Keterbukaan Citra.

Tabel 4.12 adalah hasil jalur independen dan perhitungan siklomatis berdasarkan gambar 4.14.

Tabel 4.12. Perhitungan Jalur Independen dan Kompleksitas Siklomatis.

Jalur independen		Perhitungan siklomatis		Jumlah jalur grafik=siklomatis
No.	Jalur	Simpul Predikat	V(G)=P+1	
1	1-2-3,4,5,6-7-25-26-27-33-35-37	10	11	[√] diterima [ ] ditolak
2	1-2-3,4,5,6-7-8-9-15-16-17-23-24-7-25-26-27-33-35-37			
3	1-2-3,4,5,6-7-8-9-10-12-13-14-9-15-16-17-23-24-7-25-26-27-33-35-37			
4	1-2-3,4,5,6-7-8-9-10-11-13-14-9-15-16-17-23-24-7-25-26-27-33-35-37			
5	1-2-3,4,5,6-7-8-9-10-12-13-14-9-15-16-17-18-20-21-22-17-23-24-7-25-26-27-33-35-37			
6	1-2-3,4,5,6-7-8-9-10-12-13-14-9-15-16-17-18-19-21-22-17-23-24-7-25-26-27-33-35-37			
7	1-2-3,4,5,6-7-8-9-10-12-13-14-9-15-16-17-18-19-21-22-17-23-24-7-25-26-27-28-30-32-33-35-37			
8	1-2-3,4,5,6-7-8-9-10-12-13-14-9-15-16-17-18-19-21-22-17-23-24-7-25-26-27-28-29-30-32-33-35-37			
9	1-2-3,4,5,6-7-8-9-10-12-13-14-9-15-16-17-18-19-21-22-17-23-24-7-25-26-27-28-30-31-32-33-35-37			
10	1-2-3,4,5,6-7-8-9-10-12-13-14-9-15-16-17-18-19-21-22-17-23-24-7-25-26-27-28-30-32-33-34-35-37			
11	1-2-3,4,5,6-7-8-9-10-			

	12-13-14-9-15-16-17-18-19-21-22-17-23-24-7-25-26-27-28-30-32-33-35-36-37			

4.2.1.12 Kesimpulan Hasil Uji *Alpha*

Berdasarkan hasil pengujian *Alpha* yang telah dilakukan dapat disimpulkan bahwa aplikasi sudah berjalan cukup maksimal, tetapi tidak menutup kemungkinan dapat terjadi kesalahan suatu saat, pada saat aplikasi digunakan, sehingga membutuhkan proses *maintenance* untuk lebih mengetahui kekurangan dari aplikasi.

4.2.2 Hasil Pengenalan Algoritma *Template matching* dan *Feature extraction*

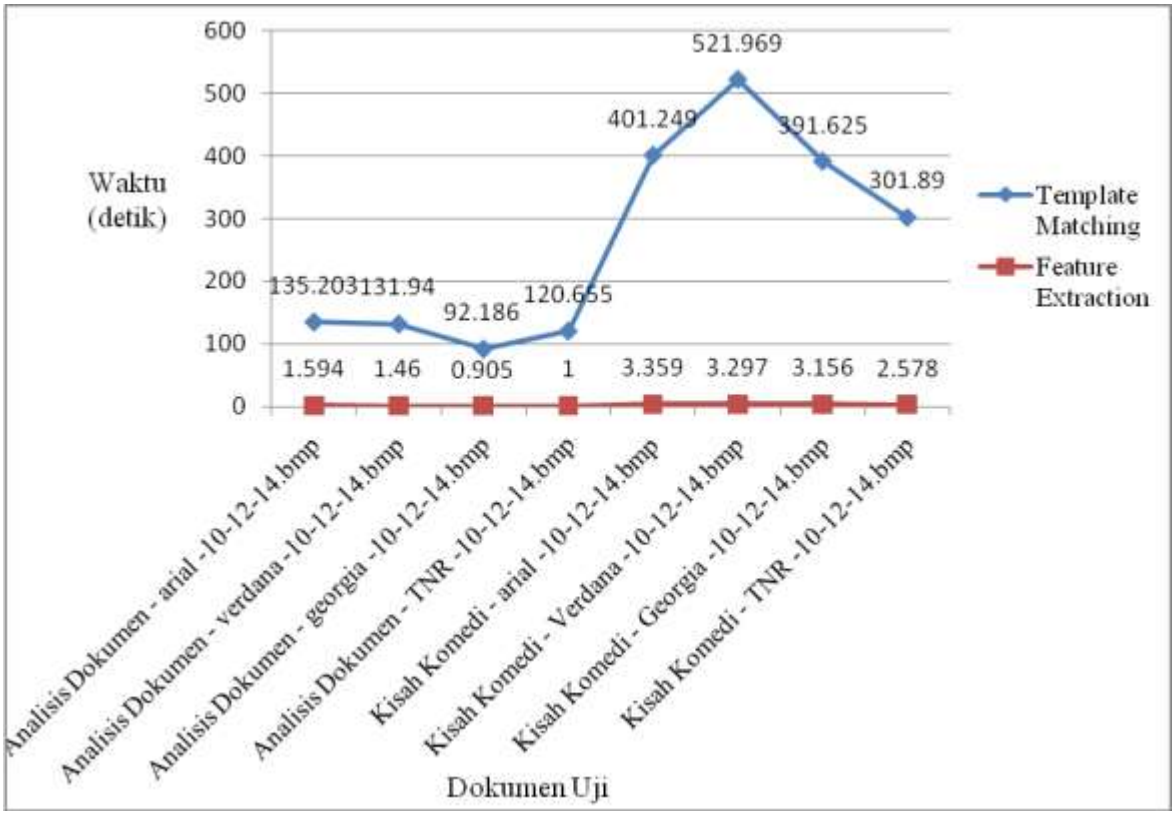
Masing-masing algoritma diuji berdasarkan persentasi hasil pengenalan dan waktu yang dibutuhkan dalam mengenali citra karakter. Algoritma-algoritma tersebut diuji dengan dokumen citra karakter digital berwarna hitam, citra karakter digital dengan kombinasi warna, citra karakter berwarna hitam hasil *scanning*, citra karkater tulisan tangan berwarna hitam. Tampilan *file* dokumen uji dan hasil pengenalan masing-masing algoritma terhadap dokumen uji terdapat pada bagian lampiran.

Pengujian pertama dilakukan terhadap citra karakter digital dengan *font* Arial, Verdana, Georgia, dan Times New Roman. Masing-masing *font* yang diujikan memiliki ukuran 10, 12, dan 14 pixel. Tabel 4.13 adalah persentase hasil pengenalan dan waktu yang dibutuhkan masing-masing algoritma terhadap masing-masing dokumen uji.

Tabel 4.13. Hasil Pengenalan Terhadap Citra Karakter Digital.

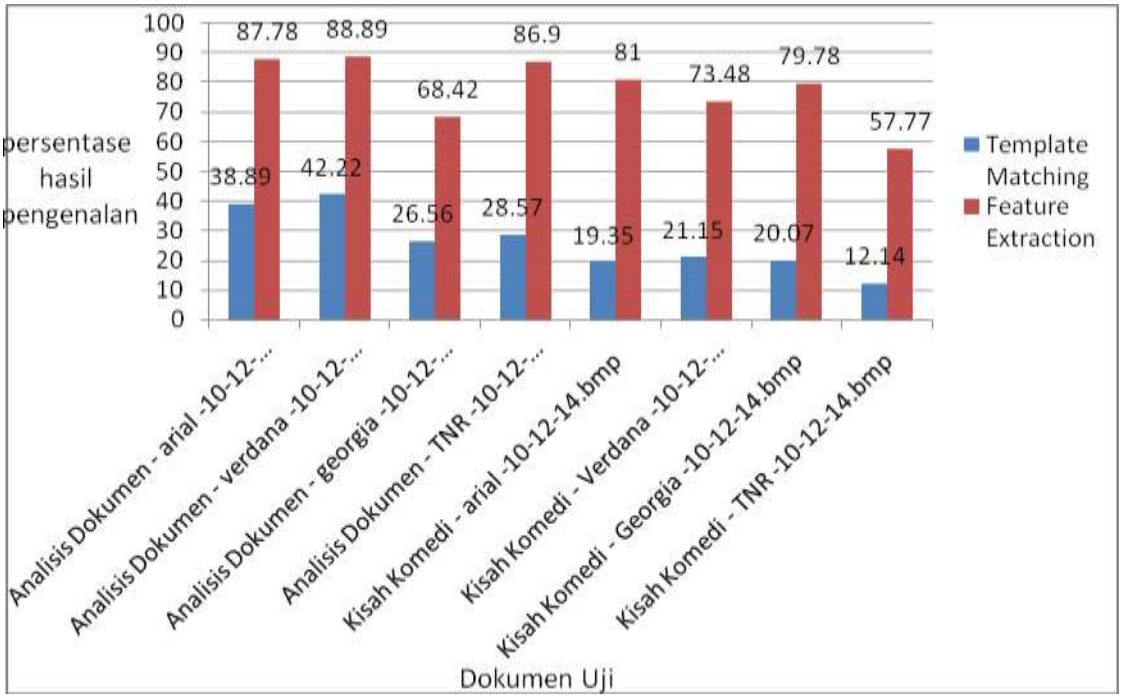
No.	Nama File	Total Karakter yang Terdeteksi	Hasil pengenalan *Template matching*		Hasil pengenalan *Feature extraction*	
			Akurasi (%)	Waktu	Akurasi (%)	Waktu
1	Analisis Dokumen - arial -10-12-14.bmp	90	38.89	2 menit 15 detik 203 ms	87.78	1 detik 594 ms
2	Analisis Dokumen - verdana -10-12-14.bmp	90	42.22	2 menit 11 detik 94 ms	88.89	1 detik 46 ms
3	Analisis Dokumen - georgia -10-12-14.bmp	57	26.56	1 menit 32 detik 186 ms	68.42	905 ms
4	Analisis Dokumen - TNR -10-12-14.bmp	84	28.57	2 menit 655 ms	86.90	1 detik 0 ms
5	Kisah Komedi - arial -10-12-14.bmp	279	19.35	6 menit 41 detik 249 ms	81.00	3 detik 359 ms
6	Kisah Komedi - Verdana -10-12-14.bmp	279	21.15	8 menit 41 detik 969 ms	73.48	3 detik 297 ms
7	Kisah Komedi - Georgia -10-12-14.bmp	272	20.07	6 menit 31 detik 625 ms	79.78	3 detik 156 ms
8	Kisah Komedi - TNR -10-12-14.bmp	206	12.14	5 menit 1 detik 890 ms	57.77	2 detik 578 ms

Gambar 4.15 adalah grafik yang menunjukan waktu yang dibutuhkan masing-masing algoritma dalam mengenali karakter pada masing-masing dokumen uji. Sumbu y adalah satuan waktu dalam detik.



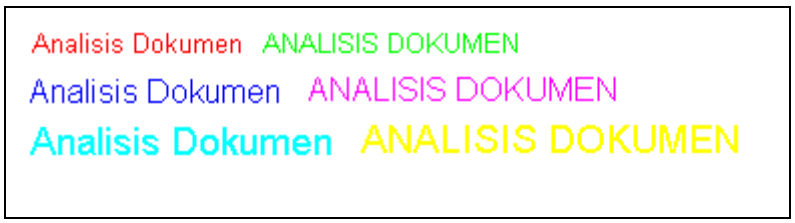
Gambar 4.15. Grafik waktu yang dibutuhkan masing-masing algoritma terhadap citra karakter digital.

Gambar 4.16 adalah grafik yang menunjukan persentase hasil pengenalan masing-masing algoritma dalam mengenali karakter digital pada setiap dokumen uji.



Gambar 4.16. Grafik Persentase Hasil Pengenalan Masing-Masing Algoritma Terhadap Citra Karakter Digital.

Pengujian kedua dilakukan terhadap citra karakter digital dengan kombinasi warna. Gambar 4.17 adalah tampilan salah satu dokumen uji dengan kombinasi warna. Warna yang digunakan pada semua dokumen uji citra karakter berwarna adalah merah, hijau, biru, ungu, *cyan*, dan kuning.



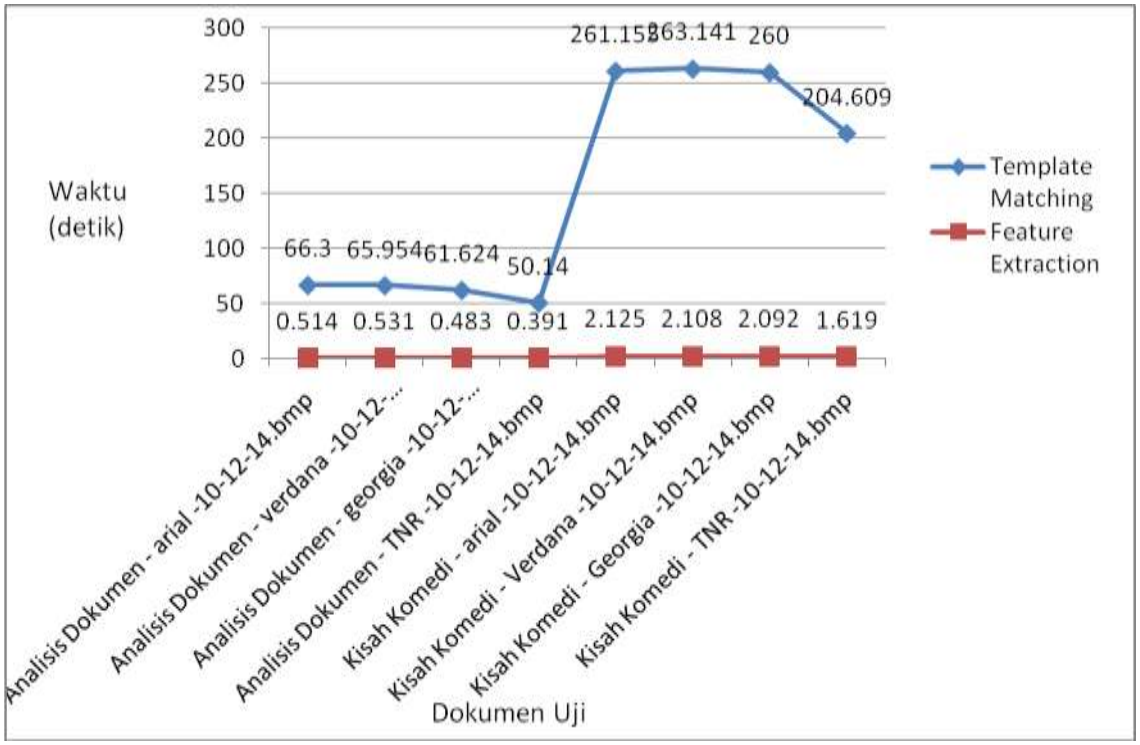
Gambar 4.17. Tampilan Salah Satu Dokumen Uji dengan Kombinasi Warna.

Dari keenam warna yang digunakan, hanya warna merah, biru, dan ungu yang bisa diproses. Tabel 4.14 adalah persentase hasil pengenalan dan waktu yang dibutuhkan masing-masing algoritma dalam mengenali dokumen citra karakter digital dengan kombinasi warna.

Tabel 4.14. Hasil Pengenalan Terhadap Citra Karakter dengan Kombinasi Warna.

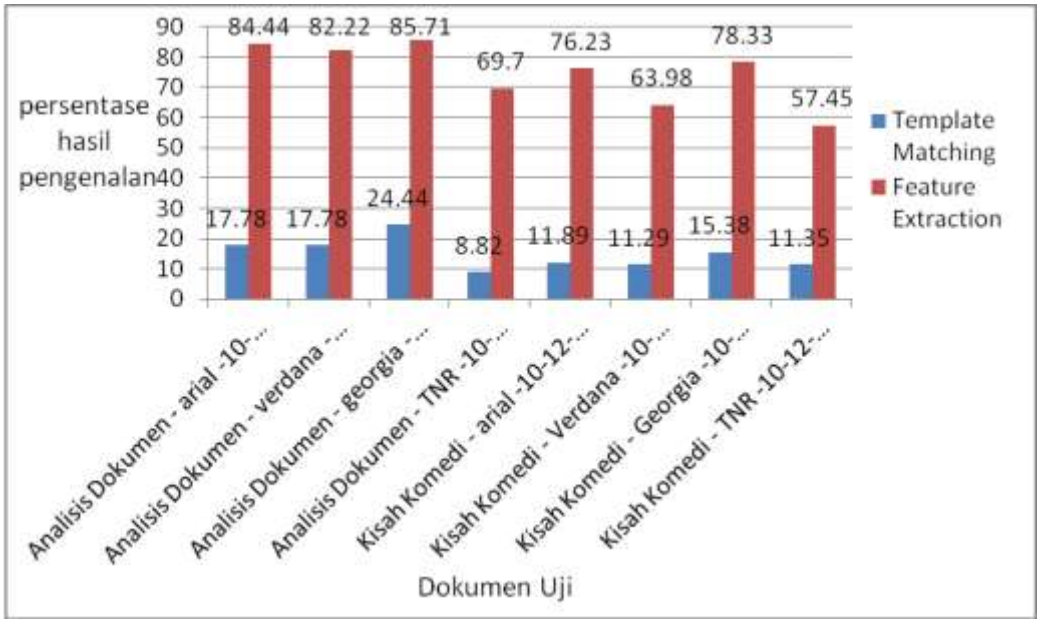
No.	Nama File	Total Karakter yang Terdeteksi	Hasil Pengenalan *Template matching*		Hasil Pengenalan *Feature extraction*	
			Akurasi (%)	Waktu	Akurasi (%)	Waktu
1	Analisis Dokumen - arial -10-12-14.bmp	45	17.78	1 menit 6 detik 30 ms	84.44	514 ms
2	Analisis Dokumen - verdana -10-12-14.bmp	45	17.78	1 menit 5 detik 954 ms	82.22	531 ms
3	Analisis Dokumen - georgia -10-12-14.bmp	42	24.44	1 menit 1 detik 624 ms	85.71	483 ms
4	Analisis Dokumen - TNR -10-12-14.bmp	33	8.82	50 detik 140 ms	69.70	391 ms
5	Kisah Komedi - arial - 10-12-14.bmp	185	11.89	4 menit 21 detik 155 ms	76.23	2 detik 125 ms
6	Kisah Komedi - Verdana -10-12-14.bmp	185	11.29	4 menit 23 detik 141 ms	63.98	2 detik 108 ms
7	Kisah Komedi - Georgia -10-12-14.bmp	180	15.38	4 menit 20 detik	78.33	2 detik 92 ms
8	Kisah Komedi - TNR - 10-12-14.bmp	141	11.35	3 menit 24 detik 609 ms	57.45	1 detik 619 ms

Gambar 4.18 adalah grafik yang menunjukan waktu yang dibutuhkan masing-masing algoritma dalam mengenali karakter digital dengan kombinasi warna pada masing-masing dokumen uji. Sumbu y adalah satuan waktu dalam detik.



Gambar 4.18. Grafik Waktu yang Dibutuhkan Masing-Masing Algoritma Terhadap Citra Karakter Digital dengan Kombinasi Warna.

Gambar 4.19 adalah grafik yang menunjukan persentase hasil pengenalan masing-masing algoritma dalam mengenali karakter digital dengan kombinasi warna pada setiap dokumen uji.



Gambar 4.19. Grafik Persentase Hasil Pengenalan Masing-Masing Algoritma Terhadap Citra Karakter Digital dengan Kombinasi Warna.

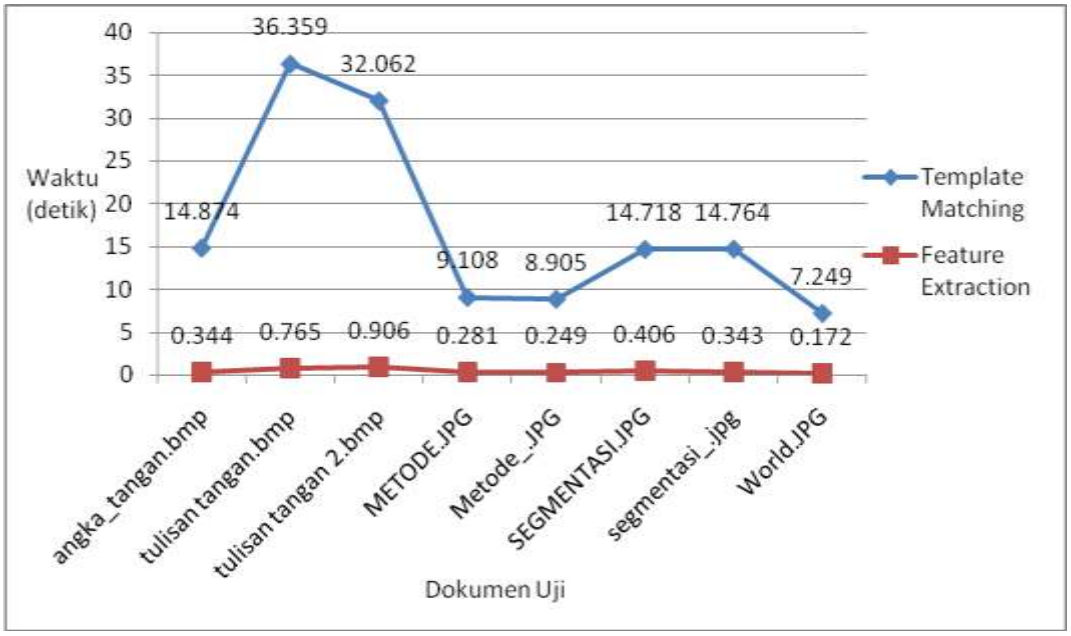
Pengujian ketiga dilakukan terhadap citra karakter tulisan tangan. Tabel 4.15 adalah persentasi hasil pengenalan dan waktu yang dibutuhkan masing-masing algoritma dalam mengenali karakter.

Tabel 4.15. Hasil Pengenalan Terhadap Citra Karakter Tulisan Tangan.

No.	Nama File	Total Karakter yang terdeteksi	Hasil Pengenalan *Template matching*		Hasil Pengenalan *Feature extraction*	
			Akurasi (%)	Waktu	Akurasi (%)	Waktu
1	angka_tangan.bmp	10	20.00	14 detik 874 ms	0.00	344 ms
2	tulisan tangan.bmp	25	16.00	36 detik 359 ms	24.00	765 ms
3	tulisan tangan 2.bmp	22	27.28	32 detik 62 ms	45.46	906 ms
4	METODE.JPG	6	33.33	9 detik 108 ms	50	281 ms
5	Metode_.JPG	6	66.67	8 detik 905 ms	33.33	249 ms
6	SEGMENTASI.JPG	10	20.00	14 detik 718 ms	50.00	406 ms

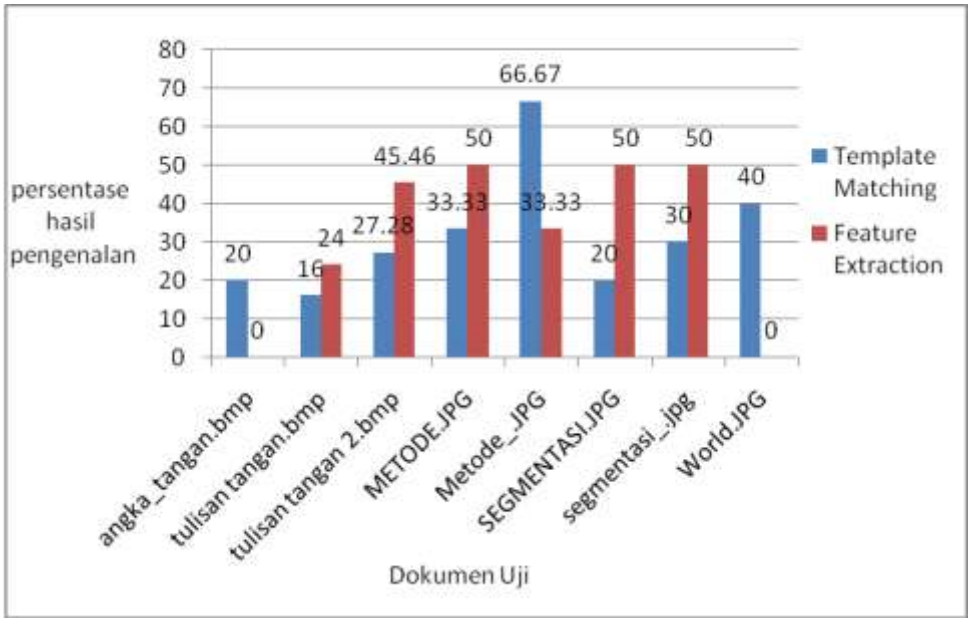
No.	Nama File	Total Karakter yang terdeteksi	Hasil Pengenalan *Template matching*		Hasil Pengenalan *Feature extraction*	
			Akurasi (%)	Waktu	Akurasi (%)	Waktu
7	segmentasi_.jpg	10	30.00	14 detik 734 ms	50	343 ms
8	World.JPG	5	40.00	7 detik 249 ms	0.00	172 ms

Gambar 4.20 adalah grafik yang menunjukan waktu yang dibutuhkan masing-masing algoritma dalam mengenali karakter tuisan tangan pada masing-masing dokumen uji. Sumbu y adalah satuan waktu dalam detik.



Gambar 4.20. Grafik Waktu yang Dibutuhkan Masing-Masing Algoritma Terhadap Citra Karakter Tulisan Tangan.

Gambar 4.21 adalah grafik yang menunjukan persentase hasil pengenalan masing-masing algoritma dalam mengenali karakter tulisan tangan pada setiap dokumen uji.



Gambar 4.21. Grafik Persentase Hasil Pengenalan Masing-Masing Algoritma Terhadap Citra Karakter Tulisan Tangan.

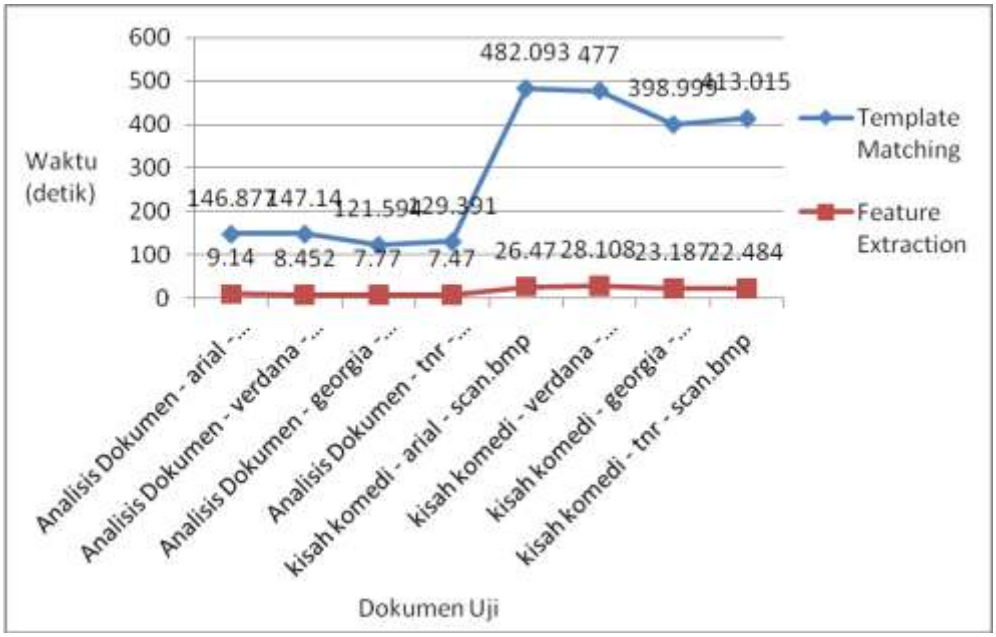
Pengujian keempat dilakukan terhadap citra karakter digital yang di-*scan* dengan resolusi 200 ppi. Tabel 4.16 adalah persentasi hasil pengenalan dan waktu yang dibutuhkan masing-masing algoritma dalam mengenali karakter digital hasil *scanning*.

Tabel 4.16. Hasil Pengenalan Terhadap Citra Karakter Digital Hasil *Scanning*.

No.	Nama File	Total Karakter yang terdeteksi	Hasil Pengenalan *Template matching*		Hasil Pengenalan *Feature extraction*	
			Akurasi (%)	Waktu	Akurasi (%)	Waktu
1	Analisis Dokumen - arial - *scan*.bmp	90	44.44	2 menit 26 detik 827 ms	67.78	9 detik 140 ms
2	Analisis Dokumen - verdana - *scan*.bmp	90	40.00	2 menit 27 detik 14 ms	76.67	8 detik 452 ms
3	Analisis Dokumen - georgia - *scan*.bmp	58	28.78	2 menit 1 detik 594 ms	91.38	7 detik 77 ms

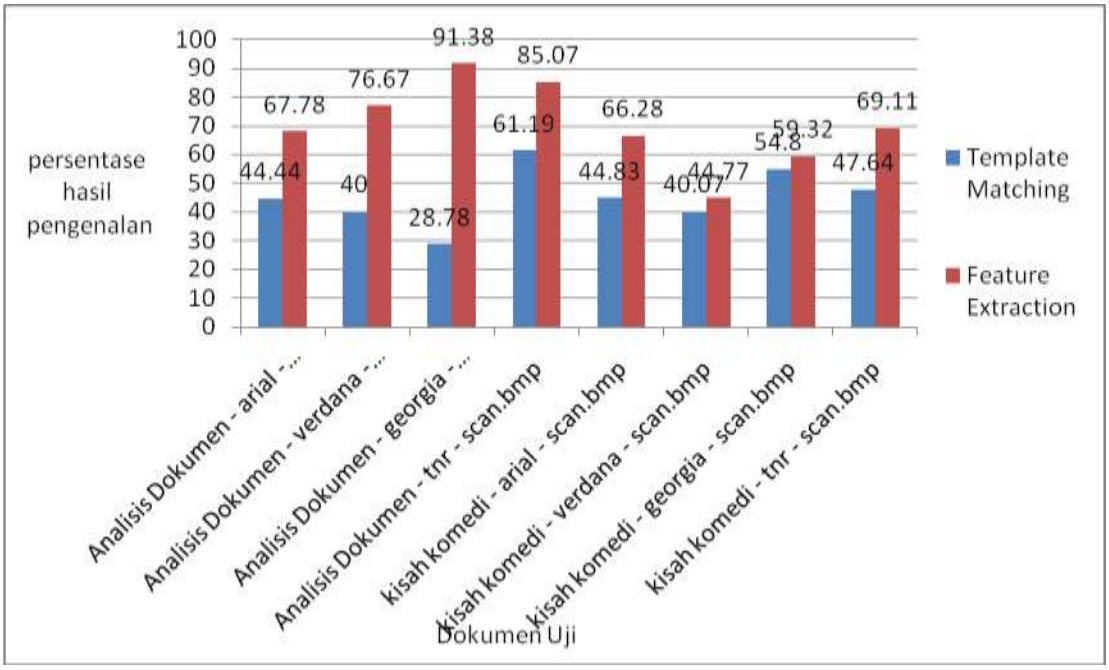
No.	Nama File	Total Karakter yang terdeteksi	Hasil Pengenalan *Template matching*		Hasil Pengenalan *Feature extraction*	
			Akurasi (%)	Waktu	Akurasi (%)	Waktu
4	Analisis Dokumen - tnr - *scan*.bmp	67	61.19	2 menit 9 detik 391 ms	85.07	7 detik 47 ms
5	kisah komedi - arial - *scan*.bmp	261	44.83	7 menit 42 detik 93 ms	66.28	26 detik 47 ms
6	kisah komedi - verdana - *scan*.bmp	277	40.07	7 menit 57 detik	44.77	28 detik 108 ms
7	kisah komedi - georgia - *scan*.bmp	177	54.80	6 menit 38 detik 999 ms	59.32	23 detik 187 ms
8	kisah komedi - tnr - *scan*.bmp	191	47.64	6 menit 53 detik 15 ms	69.11	22 detik 484 ms

Gambar 4.22 adalah grafik yang menunjukan waktu yang dibutuhkan masing-masing algoritma dalam mengenali karakter hasil *scanning* pada masing-masing dokumen uji. Sumbu y adalah satuan waktu dalam detik.



Gambar 4.22. Grafik Waktu yang Dibutuhkan Masing-Masing Algoritma Terhadap Citra Karakter Hasil *Scanning*

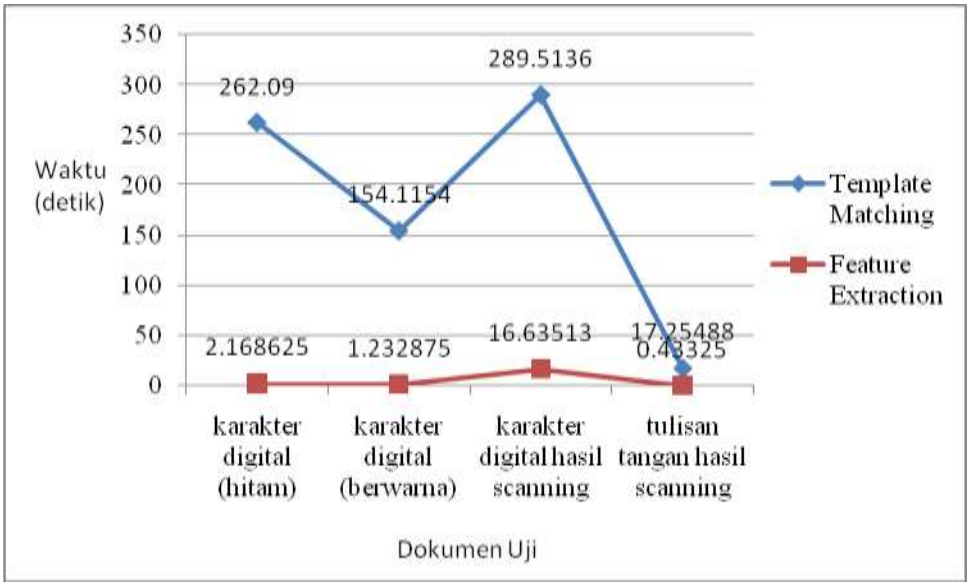
Gambar 4.23 adalah grafik yang menunjukan persentase hasil pengenalan masing-masing algoritma dalam mengenali karakter hasil *scanning* pada setiap dokumen uji.



Gambar 4.23. Grafik Persentase Hasil Pengenalan Masing-Masing Algoritma Terhadap Citra Karakter Hasil *Scanning*.

4.2.2.1 Analisis Hasil Pengenalan Algoritma *Template matching* dan *Feature extraction*

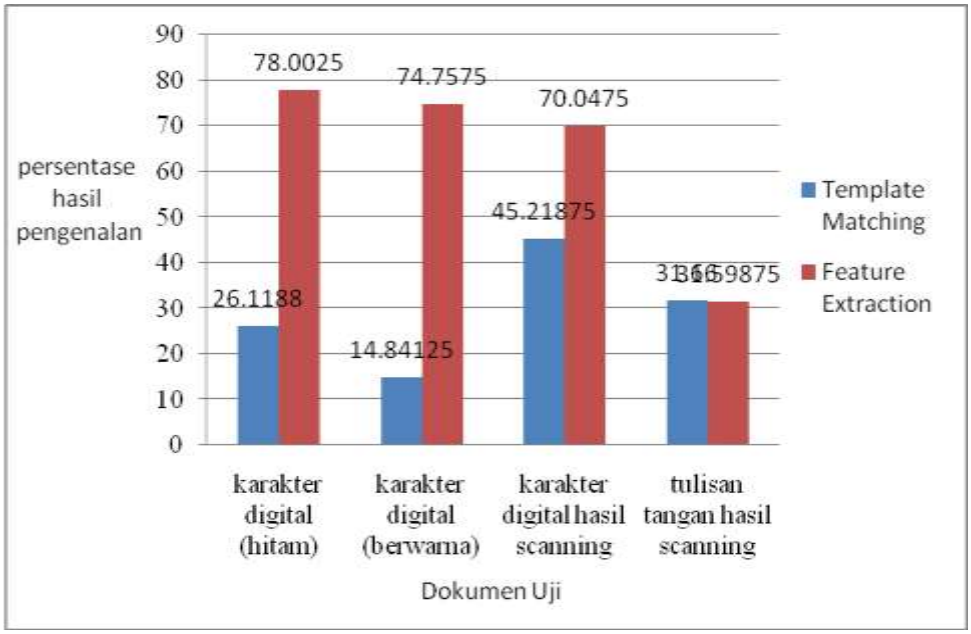
Gambar 4.24 adalah grafik waktu yang dibutuhkan masing-masing algoritma terhadap masing-masing tipe dokumen uji.



Gambar 4.24. Grafik Waktu Masing-Masing Tipe Dokumen Uji.

Berdasarkan grafik garis waktu pada gambar 4.24, grafik waktu algoritma *template matching* dengan garis yang berwarna biru, selalu berada di atas grafik waktu algoritma *feature extraction* dengan garis berwarna merah.

Gambar 4.25 adalah grafik batang persentase hasil pengenalan masing-masing algoritma terhadap masing-masing tipe dokumen uji.



Gambar 4.25. Grafik Batang Persentase Pengenalan Masing-Masing Tipe Dokumen Uji.

Berdasarkan grafik batang persentase hasil pengenalan masing-masing algoritma yang terbentuk pada gambar 4.25, algoritma *template matching* memiliki persentase pengenalan yang lebih rendah dibandingkan algoritma *feature extraction*. Namun, pada grafik hasil pengenalan masing-masing algoritma terhadap citra karakter tulisan tangan, terdapat grafik menunjukkan algoritma *template matching* lebih baik. Hal ini terjadi karena karakter tulisan tangan yang diujikan memiliki tingkat kesamaan pola pixel yang tinggi dengan pola pixel citra karakter *template* pada basis data.

4.2.2.2 Kesimpulan Hasil Pengenalan Algoritma *Template matching* dan *Feature extraction*

Berdasarkan persentasi hasil pengenalan, algoritma *Feature extraction* mampu mengenali citra karakter lebih baik dibandingkan algoritma *Template matching*. Berdasarkan waktu yang dibutuhkan masing-masing algoritma dalam mengenali citra karakter, algoritma *feature extraction* membutuhkan waktu yang

lebih cepat dalam mengenali citra karakter dibandingkan algoritma *template matching*.

BAB 5
KESIMPULAN DAN SARAN

Kesimpulan dan saran diambil berdasarkan studi pustaka, analisis, dan implementasi terhadap sistem yang dibangun.

5.1 Kesimpulan

Berikut ini adalah beberapa kesimpulan yang bisa didapat dari perbandingan algoritma *template matching* dan *feature extraction* pada OCR yang dibangun.

1. Berdasarkan hasil pengujian, algoritma *feature extraction* memiliki tingkat akurasi yang lebih baik daripada algoritma *template matching*. Algoritma *template matching* sulit dikembangkan, karena algoritma template matching bekerja dengan menyesuaikan intensitas pixel. Algoritma *feature extraction* memiliki peluang untuk bisa dikembangkan terutama pada ciri-ciri khusus citra karakter. Salah satu contohnya adalah fitur stroke (jumlah garis yang membentuk karakter). Berdasarkan proses yang dibutuhkan kedua algoritma, algoritma *feature extraction* membutuhkan proses yang lebih banyak dibandingkan dengan algoritma *template matching*. Berdasarkan hasil perhitungan kompleksitas kedua algoritma, algoritma *feature extraction* dan algoritma *template matching* memiliki kompleksitas yang sama yaitu 2n+1. Berdasarkan hasil pengujian, algoritma *template matching* membutuhkan waktu yang lebih lama dibandingkan algoritma *feature extraction* dalam mengenali dokumen uji.

2. Algoritma *feature extraction* lebih baik digunakan pada OCR dibandingkan algoritma *template matching*.

5.2 Saran

Berikut ini beberapa saran yang diajukan dalam menggunakan atau mengembangkan aplikasi:

1. Gunakan citra masukan dengan karakter-karakter yang tidak saling menyambung.
2. *Scanning* citra karakter sebaiknya menggunakan resolusi 200 ppi.
3. Gunakan citra karakter dengan *font* berwarna hitam dan latar belakang berwarna putih.
4. Untuk pengembangan, gunakan metode segmentasi yang mampu memisahkan karakter-karakter yang saling menyambung.
5. Untuk pengembangan, sebaiknya ditambahkan fitur jumlah *stroke* karakter yang banyak digunakan pada pengenalan tulisan tangan.

DAFTAR PUSTAKA

[1] A. Saeed, "Implementation of Optical Character Recognition for Mobile Phones ", Engineering Department LANCASTER UNIVERSITY, 2008.

[2] G. X. Ritter and J. N. Wilson, *Handbook of Computer Vision Algorithms in Image Algebra*: CRC Press 1996.

[3] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Second ed. San Fransisco: Elsevier, 2006.

[4] M. Adri, "Computer Vision Basic Concept," pp. 4-6, 4/4 2009.

[5] M. CHERIET, *et al., Character Recognition Systems.* New Jersey: John Wiley & Sons, 2007.

[6] M. S. Nixon and A. S. Aguado, *Feature Extraction and Image Processing*, First ed. London: Newnes, 2002.

[7] Novhard. (2007, 5/6/2011). *Pattern Recognition atau Pengenalan Pola*. Available: http://novhard.wordpress.com/2007/09/07/pattern-recognition-atau-pengenalan-pola/

[8] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Second ed. New Jersey: Prentice-Hall, 2002.

[9] R. Munir, "Kompleksitas Algoritma," Bandung, 2009.

[10] S. M. C. Y. Suen and K. Yamamoto, "Historical Review of OCR Research and Development," *IEEE*, vol. 80, p. 1031, 1992.