BAB 3
ANALISIS DAN PERANCANGAN

3.1 Analisis Masalah

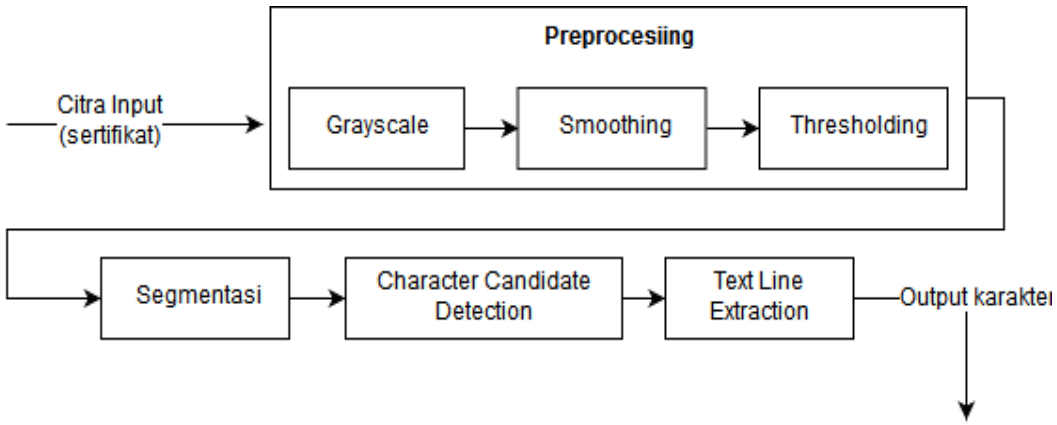
Berdasarkan latar belakang dan identifikasi masalah, maka akan dijelaskan tentang analisis masalah yang ditemukan di dalam penelitian ini. Adapun masalahnya yaitu bagaimana cara mendeteksi karakter yang ada pada sertifikat karena pada setiap sertifikat tidak hanya terdapat huruf namun juga gambar latar yang berbeda-beda yang terkadang berhimpit dengan teks penting yang terdapat pada sertifikat. Metode yang digunakan untuk klasifikasi pada teks sertifikat tersebut adalah *Text Flow*. Pada penelitian sebelumnya *Text Flow* sudah diimplementasikan pada pengenalan tulisan pada *natural scene*. Metode tersebut mempunyai hasil yang cukup baik dari tingkat akurasinya. Oleh karena itu, pada penelitian ini metode *Text Flow* digunakan sebagai pengenalan teks pada sertifikat.

3.2 Analisis Sistem

Dalam analisis sistem ini dilakukan proses penguraian konsep ke dalam bagian-bagian yang lebih sederhana, sehingga struktur logisnya menjadi jelas. Analisis merupakan metode untuk menguji, menilai dan memahami sistem pemikiran yang kompleks dengan memecahkannya ke dalam unsur-unsur yang lebih sederhana sehingga hubungan antar unsur menjadi lebih jelas.

Sistem yang dibangun memiliki masukan, proses, dan keluaran. Masukan yang terdiri dari data latih dan data uji, data latih berupa gambar yang berisi kumpulan karakter dari berbagai huruf sedangkan data uji berupa gambar hasil *scan* sertifikat. Selanjutnya untuk proses terdiri dari 3 bagian, yaitu *preprocessing*, pendeteksian kandidat karakter, serta *Minimum Flow Cost Network*. *Preprocessing* terdiri *grayscale*, *binary/thresholding* dimana hasil dari *preprocessing* tersebut nantinya akan menjadi nilai input bagi tahap pendeteksian kandidat karakter menggunakan algoritma Adaboost. Hasil pendeteksian kandidat karakter akan menjadi masukan untuk tahap *Minimum Flow Cost Network.* Setelah didapatkan baris karakter dari metode tahap *Minimum Flow*

Cost Network maka akan menampilkan keluaran berupa karakter sesuai dengan yang ada pada citra inputnya. Berikut adalah proses yang terjadi pada sistem dapat dilihat pada gambar 3.1.



Gambar 3.1 Block Diagram Proses Utama Sistem

3.3 Analisis Data Masukan

Pada sistem, data masukan tersebut merupakan suatu citra. Data masukan merupakan citra input berupa hasil *scan* sertifikat yang digunakan sebagai data uji. Berikut ini contoh data masukan pada gambar 3.2.

Berikut ini adalah spesifikasi dari data latih yang akan diolah oleh sistem:

1. Format dokumen adalah .jpg atau .png.
2. Ukuran menyesuaikan dengan citra tersebut.
3. Warna tulisan harus lebih gelap dari latar citra tersebut.

Berikut ini adalah spesifikasi dari data uji yang akan diolah oleh sistem:

1. Format dokumen adalah .jpg atau .png.
2. Ukuran menyesuaikan dengan hasil scan dokumen tersebut.

3. Dokumen tersebut berupa hasil scan dari sebuah sertifikat acara (seminar atau workshop).
4. Warna tulisan harus lebih gelap dari warna latar sertifikatnya.

Gambar 3.2 Contoh Data Uji

3.4 Analisis Metode

Pada bagian ini yaitu menjelaskan analisis metode yang terjadi pada implementasi metode *Text Flow* pada sertifikat.

3.4.1 Analisis Pendeteksian Kandidat Karakter

Pada tahapan ini, citra yang telah dimasukan akan dilakukan *preprocessing* yang terdiri dari pengubahan gambar berwarna (RGB) menjadi warna keabuan (*grayscale*), *smoothing*, dan segmentasi.

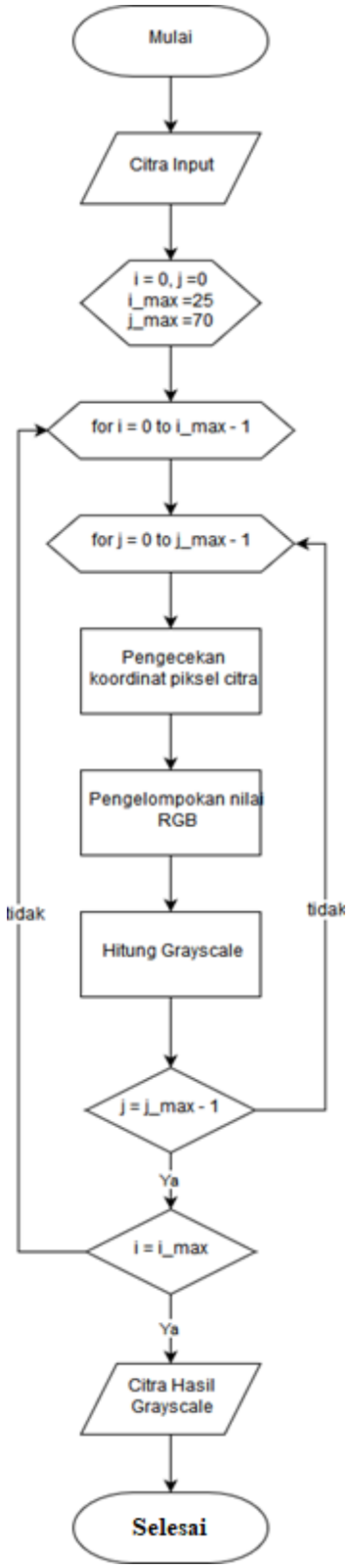
3.4.1.1 *Grayscale*

Pada tahapan ini, citra inputan yang pertama kali diterima sistem akan mengalami proses pengolahan awal yaitu proses *grayscale*. Proses *grayscale* dilakukan untuk mendapatkan citra *grayscale* atau citra dengan warna keabuan. Rumus yang digunakan untuk mengubah citra RGB menjadi citra *grayscale* adalah

"*luma*" atau "*luminance*". Berikut adalah rumus yang digunakan untuk mengubah citra menjadi *grayscale* pada persamaan 3.1 berikut.

$$Gray = 0.299 * red + 0.587 * green + 0.114 * blue \quad (3.1)$$

Alur kerja dari *grayscale* yaitu terdapat masukan berupa citra input, kemudian mengambil salah satu piksel untuk diproses, pada setiap piksel terdapat tiga nilai yang terdiri dari *red*, *green*, dan *blue*, nilai tersebut kemudian dikelompokan agar bisa diproses pada tahapan berikutnya, kemudian hitung nilai tersebut menggunakan persamaan 3.1 diatas. Setelah dihitung buatlah percabangan, percabangan pertama yaitu apabila seluruh piksel belum diproses maka ulang kembali proses tersebut, jika seluruh piksel sudah diproses maka citra telah berhasil masuk pada tahap *grayscale*. *Grayscale* memiliki satu nilai dengan *range* 0-255. Berikut ini merupakan alur dari proses *grayscale*:



Gambar 3.3 Flowchart Grayscale

Contoh perhitungan *grayscale*:

Untuk dapat melakukan perhitungan *grayscale* dibutuhkan citra masukan, citra masukan terdiri dari citra latih dan citra uji. Berikut adalah citra latih dengan ukuran 16 * 105 piksel:

Gambar 3.4 Contoh Citra Latih

Gambar 3.4 diatas merupakan contoh data latih yang nantinya akan digunakan sebagai contoh perhitungan pada setiap proses tahapan *preprocessing* maupun tahapan metode *Flow Text* itu sendiri.

Berikut adalah langkah-langkah metode *grayscale*:

1. Pertama-tama lakukan pengambilan piksel di mulai dari piksel (0,0). Citra memiliki 3 nilai warna di dalam satu pikselnya, terbentuk dari komponen *red* (merah), *green* (hijau), dan *blue* (biru). Oleh karena itu, dalam 1 piksel terdapat 3 buah nilai yang nantinya akan di proses pada tahap perhitungan.
2. Setelah melakukan pengambilan pada piksel pertama, maka kelompokkan nilai RGB tersebut agar terpisah satu sama lain.
3. Setelah dilakukan pengelompokan didapatkan nilai R = 255, G = 253, B = 255. Kemudian hitung nilai tersebut menggunakan persamaan (2.1) diatas, berikut adalah perhitungan dari piksel (0,0).

   Perhitungan *grayscale* piksel (0,0):

   *Gray = red * 0.299 + green * 0.587 + blue * 0.114*

   *Gray = 255 * 0.299 + 253 * 0.587 + 255 * 0.114*

   Gray = 76.5 + 148..511 + 28.05

   Gray = 253

4. Setelah dilakukan perhitungan pada piksel pertama, maka ulang proses tersebut hingga piksel terakhir. Berikut ini adalah hasil citra setelah mengalami proses *grayscale*:

Tabel 3.1 Nilai Citra Latih Hasil Grayscale

G(x,y)	0	1	2	3	4	5	6	⋯	104
0	253	253	252	253	255	252	249	⋯	255
1	254	251	252	245	254	254	248	⋯	255
2	247	240	213	224	255	247	253	⋯	255
3	254	159	6	92	253	254	205	⋯	255
4	253	164	3	101	255	253	218	⋯	255
5	254	152	7	95	248	252	209	⋯	255
6	252	156	18	95	207	212	169	⋯	255
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋯	⋮
15	255	255	253	247	255	255	255	⋯	255

Gambar 3.5 Contoh Citra Latih Hasil Proses *Grayscale*

Gambar 3.6 Contoh Data Uji Hasil Proses *Grayscale*

3.4.1.2 *Smoothing*

Smoothing dilakukan menggunakan citra integral dengan tujuan untuk memperbaiki kualitas citra dan menghilangkan derau. Pada tahap ini digunakan sebuah mask sebagai rentang perhitungan dengan nilai ketetanggaan.

Ukuran mask akan mempengaruhi hasil akhir dari *smoothing*. Jika ukuran mask yang digunakan berukuran kecil, maka besar kemungkinan derau yang terdapat pada citra tidak dapat dihilangkan, hal ini pun dipengaruhi ukuran dari citra dan ukuran derau yang ada. Namun jika ukuran mask terlalu besar, objek yang terdapat pada sebuah citra akan mengalami pemudaran yang berlebihan. Penting dalam menentukan ukuran mask yang tepat untuk menghasilkan citra sesuai harapan.

Data masukan dari proses *smoothing* adalah matriks *grayscale.* tahap pertama dari *smoothing* pada penelitian ini ialah pembentukan citra integral dengan menggunakan persamaan 2.4 – 2.6 pada bab 2. Sebagai contoh digunakan matriks Gx,y pada tabel 3.1 dengan menghitung citra integral pada koordinat (0,0), (0,1), (1.0) dan (1,1)

1.4.1.2.1 Pembentukan Matriks Citra Integral $(g_{x,y})$

Pemberntukan citra integral merupakan tahap awal dari *smoothing* pada penelitian ini. Berikut adalah tahapan dari pembentukan citra integral.

Tahap awal adalah pengisian nilai *cell matriks* $(g_{x,y})$ pada koordinat(0,0) dengan nilai cell pada matriks $(G_{0,0})$

$g_{0,0} = G_{0,0} = 253$

Dari proses tersebut dihasilkan kondisi matriks $g_{x,y}$ dengan *cell* pada koordinat (0,0) bernilai 253 sebagai berikut.

Tabel 3.2 Tahap awal Pembentukan Matriks $g_{x,y}$

$g_{x,y}$	0	1	⋯	104
0	253		⋯	
1			⋯	
⋮	⋮	⋮	⋱	⋮
15			⋯	

Selanjutnya perhitungan nilai matriks pada posisi baris pertama $(g_{0,y})$ menggunakan persamaan 2.4 untuk mengisi nilai cell yang masih kosong pada rentang baris. Berikut adalah contoh perhitungan pada koordinat (0,1).

$g_{0,y} = G_{0,y} + g_{0,y-1}$

$$g_{0,1} = G_{0,1} + g_{0,1-1}$$
$$= 253 + 253$$
$$= 506$$

Perhitungan tersebut dilanjutkan hingga koordinat (0,104), yaitu cell terakhir pada baris pertama matriks gx,. Hasil akhir dari proses ini dapat dilihat pada tabel 3.3 berikut.

Tabel 3.3 Tahap kedua Pembentukan Matriks $g_{x,y}$

$g_{x,y}$	0	1	2	3	4	5	6	⋯	104
0	253	506	758	1011	1266	1518	1767	⋯	26407
1								⋯	
2								⋯	
3								⋯	
4								⋯	
5								⋯	
6								⋯	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
15								⋯	

Selanjutnya adalah perhitungan nilai pada kolom pertama matriks $g_{x,y}$ (1,0) dengan menggunakan persamaan (2.5), dilajut hinggal *cell* terakhir pada rentang kolom pertama yaitu (15,0). Hasil akhir proses ini dapat dilihan pada tabel 3.4.

$$g_{x,y} = G_{x,0} + g_{x-1,0}$$
$$g_{1,0} = G_{1,0} + g_{1-1,0}$$
$$= G_{1,0} + g_{0,0}$$
$$= 253 + 253 = 506$$

Tabel 3.4 Tahap ketiga Pembentukan Matriks $g_{x,y}$

$g_{x,y}$	0	1	2	3	4	5	6	⋯	104
0	253	506	758	1011	1266	1518	1767	⋯	26407
1	506							⋯	
2	760							⋯	
3	1007							⋯	
4	1261							⋯	
5	1514							⋯	
6	1768							⋯	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
15	4045							⋯	

Tahap terakhir adalah perhitungan nilai *cell* matriks $g_{x,y}$ pada rentang baris dan kolom dimulai dari koordinat (1,1) sampai koordinat (15,104) menggunakan persamaan (2.6) tanpa menghitung kembali nilai *cell*

pada baris dan kolom pertama. Berikut contoh perhitungan pada koordinat (1,1).

$$g_{x,y} = G_{x,y} + g_{x,y-1} + g_{x-1,y} - g_{x-1,y-1}$$

$$g_{1,1} = G_{1,1} + g_{1,1-1} + g_{1-1,1} - g_{1-1,1-1}$$

$$= G_{1,1} + g_{1,0} + g_{0,1} - g_{0,0}$$

$$= 251 + 506 + 506 - 253 \ = 1010$$

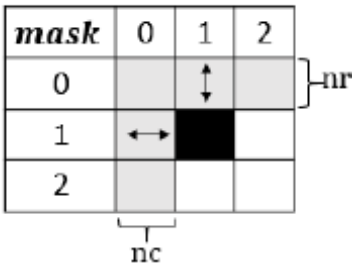
Tabel 3.5 Tahap Terakhir Pembentukan Matriks $g_{x,y}$

$g_{x,y}$	0	1	2	3	4	5	6	⋯	104
0	253	506	758	1011	1266	1518	1767	⋯	26407
1	506	1010	1514	2012	2521	3027	3524	⋯	52684
2	760	1504	2221	2943	3707	4460	5210	⋯	77537
3	1007	1910	2633	3447	4464	5471	6426	⋯	92615
4	1261	2328	3054	3969	5241	6501	7674	⋯	107741
5	1514	2733	3466	4476	5996	7508	8890	⋯	123581
6	1768	3143	3894	4999	6726	8450	10001	⋯	136574
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
15	4045	7107	8990	11640	15612	19569	23150	⋯	327436

1.4.1.2.2 Perhitungan Rata-rata Dari Nilai Pembentukan Matriks $g_{x,y}$

Setelah citra integral terbentuk, tahapan selanjutnya adalah perhitungan rata-rata nilai *cell* dari matriks $g_{x,y}$. Pada tahap ini diperlukan sebuah *window*/ dimensi *mask* pada koordinat baris dan kolom ($wx \times wy$) dengan ukuran (3 x 3). Berikut adalah tahapan dari perhitungan rata-rata nilai *cell* matriks integral menggunakan koordinat (0,0).

1. Pencarian jarak terluar mask untuk memperoleh jumlah baris (nr) dan kolom (nc) terhadap titik pusatnya. Pada mask berukuran (3×3), titik pusat dari mask barada pada koordinat (1,1). Sebagai ilustrasi dapat dilihat pada Gambar 3.7 berikut.



Gambar 3.7 ilustrasi *nc* dan *nr*

$$nr = trunc\ (\frac{wx}{2})$$
$$= trunc\ (\frac{3}{2})$$
$$= trunc\ (1.5)$$
$$= 1$$

Dikarenakan *wx* =*wy* maka nilai *nc* = *nr*

2. Menghitung jumlah baris dan kolom mask yang berada diluar matriks $g_{x,y}$ketika titik pusat mask (1,1) berada pada tepi matriks tersebut. Proses ini direpresentasikan dengan fungsi $rp(x)$ dan $cp(y)$. Dalam menentukan rumus $rp(x)$ dan $cp(y)$ perlu dilakukan pengecekan kondisi dengan menggunakan persamaan dengan melihat Tabel 3.6. Pada proses perhitungan selanjutnya, ukuran lebar dan tinggi matriks $g_{x,y}$ direpresentasikan dengan variabel M=16 dan N=105.

Tabel 3.6 Syarat perhitungan *rp(x)* dan *cp(y)*

No	Fungsi	Syarat	Rumus
1	$rp(x)$	$x-nr<0$	$\|x-nr\|$
		$x+nr\geq M$	$x+nr-M+1$
2	$cp(y)$	$y-nc<0$	$\|y-nc\|$
		$y+nc\geq N$	$y+nc-N+1$

Dengan koordinat (0,0) sebagai contoh serta menggunakan nr,nc=1 dari hasil perhitungan sebelumnya, rumus $rp(x)$ yang memenuhi syarat adalah $|x-nr|$ dengan pembuktian sebagai berikut.

$x-nr < 0$

$0-1 < 0$

$-1<0$

Berdasarkan persamaan 2.12, karena $x-nr$ sama dengan -1 maka $rp(x)$ = $|x-nr|$.

Sedangkan rumus (y) yang memenuhi syarat adalah |$y-nc$| dengan pembuktian sebagai berikut.

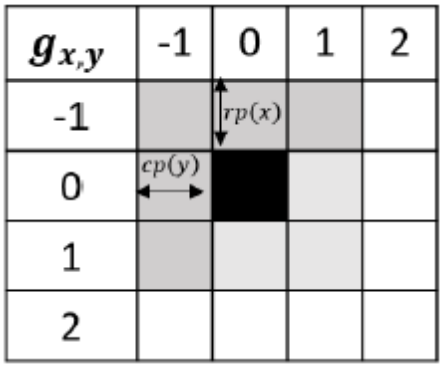
$y-nc < 0$

$0-1 < 0$

$-1 < 0$

Berdasarkan persamaan 2.12, karena $x-nr$ sama dengan -1 maka cp(y) = |$y-nc$|.

Ilustrasi dari perhitungan (x) dan (y) direpresentasikan pada Gambar 3.8 berikut.



Gambar 3.8 Ilustrasi (x) dan (y)

Dari Gambar 3.6, dapat dilihat bahwa fungsi (x) dan (y) bertujuan untuk menghitung jumlah baris dan kolom yang berada di rentang *mask* namun diluar area matriks $g_{x,y}$.

3. Menghitung total piksel berwarna putih dari *mask* yang tidak memiliki nilai. Perhitungan ini direpresentasikan dengan fungsi (x,). Dalam menentukan rumus $nw(x,y)$ perlu diperhatikan kondisi pada persamaan tabel 3.7 berikut

Tabel 3.7 Syarat perhitungan *nw(x,y)*

No	Syarat	Rumus
1	$x-nr<0$ dan $y-nc<0$ atau $x-nr<0$ dan $y+nc\geq M$ atau $x+nr\geq M$ dan $y-nc<0$	$(\boldsymbol{rp}(\boldsymbol{x})\times\boldsymbol{wy}+\lvert y-nc\rvert \times (\boldsymbol{wx}-\boldsymbol{rp}(\boldsymbol{x})))\times\boldsymbol{255}$

	atau $x+nr \geq M$ dan $y+nc \geq N$	
2	($x-nr<0$ atau $x+nr \geq M$) dan $0<y<N$	$\boldsymbol{rp(x) \times wy \times 255}$
3	($y-nc<0$ atau $y+nc \geq N$) dan $0<x<M$	$\boldsymbol{cp(y) \times wx \times 255}$

Berdasarkan tabel diatas dilakukan pengecekan terhadap setiap syarat dengan menggunakan contoh koordinat (0,0), nilai fungsi $rp(x)$ dan $cp(y)$ dari perhitungan sebelumnya. Berdasarkan kondisi maka akan dilakukan pengecekan sebagai berikut.

$x-nr < 0$

$0-1 < 0$

$-1 < 0$

dan $y-nc<0$

$0-1<0$

$-1<0$

Berdasarkan kondisi diatas maka yang terpenuhi adalah kondisi pertama. Berikut adalah contoh perhitungan $nw(x,y)$ pada koordinat(0,0).

$$nw(x,y) = (rp(x) \times wy + |y - nc| \times (wx - rp(x))) \times 255$$

$$nw(0,0) = (rp(0) \times wy + |0 - nc| \times (wx - rp(0))) \times 255$$

$$= (1 \times 3 + |0 - 1| \times (3 - 1)) \times 255$$

$$= (3 + |-1| \times 2) \times 255 = (3 + 1 \times 2) \times 255$$

$$= (3 + 2) \times 255 = 5 \times 255$$

$$= 1275$$

4. Setelah nilai dari fungsi (x,) diperoleh, selanjutnya dilakukan perhitungan $a(x,y)$. Fungsi ini bertujuan untuk menghitung rata – rata dari nilai $nw(x,y)$.

$$a(x,y) = \frac{nw(x,y)}{wx \times wy}$$

$$a(0,0) = \frac{nw(0.0)}{wx \times wy}$$

$$= \frac{1275}{3 \times 3}$$

$$= 141.6667$$

5. Selanjutnya adalah perhitungan nilai rata – rata keseluruhan menggunakan fungsi $mean(x,y)$ pada koordinat (0,0) dengan diawali perhitungan nilai ukuran *mask* dari jarak terluar hingga titik pusatnya pada rentang baris (dx) dan kolom (dy).

$$dx = round(\frac{wx}{2})$$

$$= round(\frac{3}{2})$$

$$= round(1.5)$$

$$= 2$$

Karena wx=wy maka dx=dy.

6. Setelah nilai dx dan dy diperoleh, proses dilanjutkan dengan perhitungan nilai intensitas warna pada matriks citra integral (x,y) . Untuk menghitung nilai (x,y) perlu diperhatikan kondisi – kondisi pada Tabel 3.8.

Tabel 3.8 Kondisi dan Aksi Pada Proses s(x,y)

No	Kondisi $g(x,y)$	Aksi
1	x<0 atau y<0	Nilai gx,y=0
2	$x>(M-1)$ dan $0\leq y\leq(N-1)$	Ubah koordinat x pada $g_{x,y}$ menjadi $g_{M-1,y}$
3	$0\leq x\leq(M-1)$ dan $y>(N-1)$	Ubah koordinat y pada $g_{x,y}$ menjadi $g_{x,N-1}$

4	$x>(M-1)$ dan $y>(N-1)$	Ubah koordinat x dan y pada $g_{x,y}$ menjadi $g_{M-1,N-1}$
5	$0\leq x\leq(M-1)$ dan $0\leq y\leq(N-1)$	Nilai dan koordinat pada gx,y tetap

Perhitungan ($\boldsymbol{x}$,) pada koordinat (0,0) adalah sebagai berikut.

$$s(x,y) = (g_{x+dx-1,y+dy-1} + g_{x-dx,y-dy}) - (g_{x-dx,y+dy-1} + g_{x+dx-1,y-dy})$$

$$s(0,0) = (g_{0+2-1,0+2-1} + g_{0-2,0-2}) - (g_{0-2,0+2-1} + g_{0+2-1,0-2})$$
$$= (g_{1,1} + g_{-2,-2}) - (g_{-2,1} + g_{1,-2)}$$
$$= (1010 + 0) - (0 + 0)$$
$$= 1010$$

Pada perhitungan (0,0) tersebut, berlaku kondisi ke-1 pada Tabel 3.8. Sehingga menyebabkan perubahan nilai pada koordinat (−2,−2), g(−2,1), dan g(1,−2) menjadi nol (0).

7. Menghitung nilai rata – rata $mean(x,y)$ pada koordinat (0,0) .

$$mean(x,y) = \frac{s(x,y)}{wx \times wy} + a(x,y)$$

$$mean(0,0) = \frac{s(0,0)}{wx \times wy} + a(0,0)$$
$$= \frac{1010}{3 \times 3} + 141.6667$$
$$= 112.2222 + 141.6667$$
$$= 254$$

1.4.1.2.3 Inisialisasi Nilai Rata-rata pada Matriks $sm_{x,y}$

Selanjutnya tahap terakhir dari *smoothing* adalah menginisialisasikan seluruh hasil perhitungan rata – rata terhadap matriks citra hasil *smoothing* $sm_{x,}$. Hasil dari proses ini dapat dilihat pada tabel 3.9 berikut.

Tabel 3.9 Matriks Citra Hasil Smoothing

$Sm_{x,y}$	0	1	2	3	4	5	6	⋯	104
0	254	252	251	252	252	249	249	⋯	105
1	247	252	251	252	253	253	249	⋯	255
2	240	247	224	245	253	253	248	⋯	254
3	164	213	159	213	253	253	218	⋯	255
4	159	159	95	95	252	252	209	⋯	254
5	156	156	95	95	212	218	209	⋯	253
6	156	156	95	95	210	210	178	⋯	253
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋯	⋮
15	255	253	252	252	249	249	249	⋯	255

Gambar 3.9 Contoh Data Latih Hasil Proses *Smoothing*

Gambar 3.10 Contoh Data Uji Hasil Proses *Smoothing*

3.4.1.3 *Thresholding*

Proses kedua yaitu pengolahan citra menjadi citra biner atau citra hitam putih. Citra yang diubah ke nilai biner yaitu citra output hasil dari proses *grayscale*. Cara yang digunakan untuk menghasilkan citra biner dengan melakukan perbandingan, apabila nilai piksel lebih besar dari nilai ambang batas, maka ubah menjadi nilai 0, dan apabila tidak maka nilai akan diubah menjadi 1. Berdasarkan

riset yang dilakukan oleh Siti dengan judul Analisis Nilai *Threshold* Untuk Membentuk Citra Biner Pada Citra Digital, dkk terdapat 3 nilai ambang batas di dalam pengujiannya, yaitu nilai tengah, nilai minimum, dan nilai maksimum. Nilai tengah adalah 128, nilai minimum diambil dari nilai tengah dibagi 2 sehingga nilai minimum adalah 64, sedangkan nilai maksimum adalah 128 ditambah dengan 64 sehingga hasilnya adalah 192. Penelitian tersebut menyatakan bahwa nilai tengah mempunyai hasil warna yang lebih jelas ketika dibandingkan dengan nilai minimum dan maksimum [13]. Oleh karena itu, nilai yang akan digunakan sebagai nilai ambang batas pada penelitian ini adalah 128.

Berikut adalah algoritma biner yang digunakan pada tahap *thresholding* pada persamaan 2.2 diatas:

```
if (gray > 128) {
        val = 0
} else {
        val = 1
}
```

Berikut adalah langkah-langkah metode *thresholding*:

1. Pertama-tama lakukan pengambilan piksel di mulai dari piksel (0,0). Pada tahapan *thresholding*, citra yang digunakan merupakan citra yang sudah melalui proses *grayscale*, citra hasil *grayscale* memiliki 1 nilai di setiap satu pikselnya, nilai tersebut yang nantinya akan digunakan untuk proses *thresholding*.
2. Setelah dilakukan pengambilan piksel, maka lakukan perhitungan pada nilai tersebut menggunakan persamaan (2.10) diatas. Berdasarkan gambar 3.12 diatas dapat dilihat bahwa nilai *grayscale* pada piksel (0,0) adalah 255. Nilai 255 > 128 maka nilai binernya adalah 0. Hal ini akan dilakukan hingga piksel terakhir. Berikut adalah hasil citra setelah mengalami proses *thresholding*.

Tabel 3.10 Matriks Citra Hasil Thresholding

b(x,y)	0	1	2	3	4	5	6	⋯	104
0	0	0	0	0	0	0	0	⋯	0
1	0	0	0	0	0	0	0	⋯	0
2	0	0	0	0	0	0	0	⋯	0
3	0	0	0	1	0	0	0	⋯	0
4	0	0	1	1	0	0	0	⋯	0
5	0	0	1	1	0	0	0	⋯	0
6	0	0	1	1	0	0	0	⋯	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋯	⋮
15	0	0	0	0	0	0	0	⋯	0

Gambar 3.11 Data Latih Hasil Proses *Thresholding*

Gambar 3.12 Data Uji Hasil Proses *Thresholding*

3.4.1.4 *Segmentasi*

Segmentasi pada penelitian ini digunakan untuk memisahkan objek dan menentukan objek yang dianggap sebagai karakter (kandidat karakter) pada sebuah citra masukan. Dalam hal ini, citra sudah melalui tahapan thresholding sehingga terdapat nilai satu (1) dan nol (0). Hasil keluaran dari proses segmentasi ini ialah berupa objek-objek yang ada dalam sebuah citra.

Proses segmentasi diawali dengan masuknya matriks citra biner $b_{x,y}$, selanjutnya sistem akan mencacah setiap koordinat pada matriks tersebut untuk mencari nilai intensitas satu atau berwarna putih. Jika nilai satu (1) ditemukan, maka selanjutnya sistem akan menyimpan koordinat (*x*,*y*) tersebut kedalam sebuah variabel dan mengubah nilai pada koorditan matriks $b_{x,y}$ tersebut menjadi (0).

Variabel yang digunakan untuk menampung nilai satu (1) berupa matriks yang diberi nama $pt_{i,j}$, dimana i merupakan indeks dari objek yang ditemukan dengan nilai awal nol (0). Proses dilanjutkan dengan mencari matriks dengan nilai satu (1) pada level ketetanggaan dengan *mask* berdimensi 5x5. Proses selanjutnya sistem akan mencacah nilai pada matriks $pt_{i,j}$ dari awal hingga akhir untuk dijadikan titik pusat pencarian data matriks biner. Berikut adalah proses pada tahap segmentasi menggunakan matriks biner $b_{x,y}$ pada tabel 3.11.

1. Definisikan variabel pendukung.
   Matriks untuk menyimpan koordinat objek = $pt_{i,j}$
   Matriks untuk menyimpan batas koordinat objek = $br_{i,j}$
   Array 1D untuk menyimpan nilai koordinat x dan y sementara = c
2. Pencarian dimulai dari koordinat (0,0) sampai menemukan matriks bernilai satu (1). Pada kasus ini, ditemukan matriks bernilai satu (1) pada koordinat (3,3) seperti pada tabel 3.11.

Tabel 3.11 Matriks $b_{x,y}$ Koordinat (3,3)

b(x,y)	0	1	2	3	4	5	6	⋯	104
0	0	0	0	0	0	0	0	⋯	0
1	0	0	0	0	0	0	0	⋯	0
2	0	0	0	0	0	0	0	⋯	0
3	0	0	0	1	0	0	0	⋯	0
4	0	0	1	1	0	0	0	⋯	0
5	0	0	1	1	0	0	0	⋯	0
6	0	0	1	1	0	0	0	⋯	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋯	⋮
15	0	0	0	0	0	0	0	⋯	0

3. Simpan koordinat tersebut pada matriks $pt_{i,j}$, lalu ubah nilai matriks $\boldsymbol{b_{2,3}}$ dengan nilai nol(0).

Tabel 3.12 Isi Matriks $pt_{0,j}$ Iterasi Pertama

$Pt_{0,j}$	Nilai (x,y)
$Pt_{0,0}$	3,3

4. Simpan batas koordinat objek kedalam matriks $br_{0,j}$. karena variabel tersebut saat ini belum memiliki nilai, maka nilai batas atas dan batas bawah akan diisi dengan nilai koordinat *x*, batas kiri dan batas kanan akan diisi dengan koordinat *y*, seperti pada tabel 3.13 berikut.

Tabel 3.13 Isi Matriks $br_{0,j}$ Iterasi Pertama

$br_{0,j}$	Nilai Koordinat	Keterangan
$br_{0,0}$	3	Atas
$br_{0,1}$	3	Kanan
$br_{0,2}$	3	Bawah
$br_{0,3}$	3	Kiri

5. Selanjutnya cari nilai piksel matriks tetangga bernilai satu dengan koordinat pusat (3,3) dengan *mask* berdimensi 5 X 5. Berikut adalah representasi dari proses pencarian tersebut.

Tabel 3.14 Pencarian Nilai satu Pada Piksel Tetangga

b(x,y)	0	1	2	3	4	5	6	⋯	104
0	0	0	0	0	0	0	0	⋯	0
1	0	0	0	0	0	0	0	⋯	0
2	0	0	0	0	0	0	0	⋯	0
3	0	0	0	0	0	0	0	⋯	0
4	0	0	1	1	0	0	0	⋯	0
5	0	0	1	1	0	0	0	⋯	0
6	0	0	1	1	0	0	0	⋯	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋯	⋮
15	0	0	0	0	0	0	0	⋯	0

Ditemukan piksel tetangga yang memiliki nilai (1), yaitu pada koordinat $b_{4,2}$, $b_{4,3,}$ $b_{5,2}$ dan $b_{5,3}$.

6. Simpan koordinat piksel tersebut kedalam matriks $pt_{i,j}$ lalu inisialisasikan nilai pada koordinat matriks $b_{3,2}$, $b_{3,3}$, $b_{3,4}$, $b_{4,2}$, $b_{4,3}$ dan $b_{4,4}$ dengan nilai nol (0). Berikut adalah isi dari matriks $pt_{i,j.}$

Tabel 3.15 Isi Matriks $pt_{0,j}$ Iterasi Kedua

$Pt_{0,j}$	Nilai (x,y)
$Pt_{0,0}$	3,3
$Pt_{0,1}$	4,2
$Pt_{0,2}$	4,3
$Pt_{0,3}$	5,2
$Pt_{0,4}$	5,3

7. Gunakan koordinat (4,2), inisialisasikan koordinat *x* pada variable C_0 dan koordinat *y* pada variabel C_1 . Selanjutnya lakukan perbandingan nilai pada matriks $br_{0,\ j}$ dari data sebelumnya dengan koordinat (4,2). Proses ini memperoleh nilai batas terluar untuk suatu objek.

Tabel 3.16 Perbandingan Nilai Batas Iterasi Kedua

Mencari Batas Terluar Objek ke-1			
No	**Aturan**	**Kondisi**	**Nilai**
1	$br_{0,0} > c_0$	$3 > 4$	*False*
2	$br_{0,1} > c_1$	$3 < 2$	*False*
3	$br_{0,2} > c_0$	$3 < 4$	*True*
4	$br_{0,3} > c_1$	$3 > 2$	*True*

Dari hasil perbandingan pada tabel diatas, kondisi pada nomor tiga dan empat yang terpenuhi, sehinga akan dilakukan perubahan pada matriks $br_{0,2}$ yang merepresentasikan posisi paling bawah dan $br_{0,3}$ yang merepresentasikan posisi paling kiri. Berikut adalah tabel yang berisi matriks $br_{0,j}$ iterasi kedua.

Tabel 3.17 Isi Matriks $br_{0,j}$ Iterasi Kedua

$br_{0,j}$	Nilai Koordinat	Keterangan
$br_{0,0}$	3	Atas
$br_{0,1}$	3	Kanan
$br_{0,2}$	4	Bawah
$br_{0,3}$	2	Kiri

Pada tahapan selanjutnya, proses memasuki iterasi selanjutnya dengan menggunakan koordinat $b_{4,2}$. Proses pada langkah nomor 5 sampai 7 akan diulang dengan menggunakan koordinat selanjutnya pada matriks $pt_{0,j}$ hingga tidak ada lagi koordinat yang dapat diproses pada matriks tersebut.

Jika sudah tidak ada koordinat yang dapat diproses, selanjutnya sistem akan melakukan pencacahan dengan melanjutkan proses dari koordinat $b_{3,2}$ pada tahapan nomor dua. Proses ini dilanjutkan untuk mencari objek lainnya yang mungkin masih ada pada citra masukan.

Hasil akhir dari contoh ini akan mendeteksi sebuah citra dan menyimpan setiap koordinat piksel berwarna putih kedalam matriks $pt_{i,j}$ dan menghasilkan batas tepi setiap objek yang terdeteksi pada citra masukan dan disimpan pada matriks $br_{i,j.}$ Berikut adalah tabel yang terdapat nilai pada matriks $br_{0,j}$.

Tabel 3.18 Isi Matriks $br_{0,j}$ Tahap Akhir Proses

$br_{0,j}$	Nilai Koordinat	Keterangan
$br_{0,0}$	3	Atas
$br_{0,1}$	3	Kanan
$br_{0,2}$	11	Bawah
$br_{0,3}$	2	Kiri

Data koordinat piksel berwarna putih dapat dilihat pada tabel berikut.

Tabel 3.19 Isi Matriks $pt_{o,j}$ Tahap Akhir Proses

$Pt_{0,j}$	Nilai (x,y)
$Pt_{0,0}$	3,3
$Pt_{0,1}$	4,2
$Pt_{0,2}$	4,3
$Pt_{0,3}$	5,2
$Pt_{0,4}$	5,3
⋮	⋮
$Pt_{0,16}$	11,3

Nilai koordinat yang terdapat pada tabel diatas akan digunakan untuk tahapan selanjutnya.

Tabel 3.20 Contoh hasil dari proses segmentasi

x/y	0	1	2	3	4
3	0	0	0	1	0
4	0	0	1	1	0
5	0	0	1	1	0
6	0	0	1	1	0
7	0	0	1	1	0
8	0	0	1	1	0
9	0	0	1	1	0
10	0	0	1	1	0
11	0	0	0	1	0

3.4.2 Analisis *Min-Cost Flow Netwok*

Pada tahap ini akan dibahas perancangan, penentuan *cost* dari setiap calon kandidat karakter dan implementasi dari *Min-Cost Flow Network*. Calon karakter yang terdeteksi oleh proses segmentasi akan diproses oleh tesseract sebelum memasuki tahapan *Min-Cost Fow Network*, hal ini dilakukan untuk mengecek setiap calon karakter, apakah meupakan sebuah karakter atau bukan. Jika hasil dari tesseract menyatakan sebuah calon karakter adalah benar karakter, maka karakter

tersebut akan ditandai dengan kotak berwana hijau, sedangkan jika dianggap bukan karakter maka akan ditandai dengan kotak berwarna merah. Selanjutnya kotak yang berwarna hijau lah yang akan diproses untuk proses *Min-Cost Flow Network.* Berikut adalah contoh hasil segmentasi dan proses pengenalan karakter dengan menggunakan *tesseract* pada gambar 3.13 dibawah.

Gambar 3.13 Contoh Data Uji Hasil Segmentasi

2.4.1.1 Analisis Perancangan *Min-Cost Flow Network*

Sebelum pembentukan teks akan dilakukan dulu pengecekan, apakah citra input memiliki lebih dari satu baris calon teks atau tidak. Pengecekan dilakukan berdasarkan sumbu Y pada karakter hasil segmentasi pada proses sebelumnya. Berikut adalah contoh untuk melakukan pengecekan baris dengan contoh pada Tabel 3.21.

Tabel 3.21 Calon Karakter Kandidat I sebagai A (kotak merah) dan Calon Karakter Kandidat N sebagai B (kotak kuning)

x/y	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
3	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
4	0	0	1	1	0	0	0	1	1	1	0	0	1	1	0
5	0	0	1	1	0	0	0	1	1	1	0	0	1	1	0
6	0	0	1	1	0	0	0	1	1	1	1	1	1	1	0
7	0	0	1	1	0	0	0	1	1	1	1	1	1	1	0
8	0	0	1	1	0	0	0	1	1	1	1	1	1	1	0
9	0	0	1	1	0	0	0	1	1	0	0	1	1	1	0
10	0	0	1	1	0	0	0	1	1	0	0	1	1	1	0
11	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0

Dari tabel dapat dimisalkan bahwa b1 ditandai dengan kotak berwarna merah dan b2 ditandai dengan kotak berwarna kuning.

a. Keterangan b1

b1.y = 3

b1.h = 9

b. Keterangan b2

b2.y = 3

b2.h = 9

Berikut ini adalah fungsi yang digunakan untuk melakukan ppengecekan baris.

```
Box boxA; Box boxB;
if (b1.y > b2.y) {
                    boxA = b2; boxB = b1;
            } else {
```

```
            boxA = b1; boxB = b2;
        }
        return (boxB.y >= boxA.y && boxB.y <= ( boxA.y + boxA.h)) || (((boxB.y + boxB.h) >= boxA.y && (boxB.y + boxB.h) <= (boxA.y + boxA.h)));
```

keterangan:

b1 = karakter 1

b2 = karakter 2

b1.y = koordinat y dari karakter 1

b2.y = koordinat y dari karakter 2

jika fungsi ini menghasilkan nilai *true* maka kedua objek yang masuk kedalam fungsi akan dianggap sebagai satu baris, jika tidak maka akan dianggap sebagai baris yang berbeda.

Berdasarkan tabel Tabel 3.21 maka pengecekan baris adalah sebagai berikut.

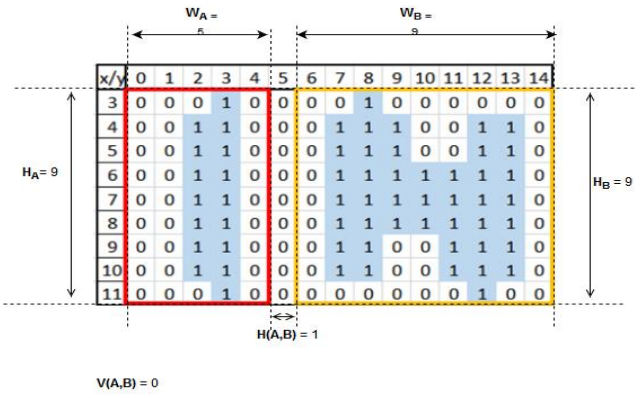
```
Box boxA; Box boxB;
if (3 > 3) {
            boxA = b2; boxB = b1;
        } else {
            boxA = b1; boxB = b2;
        }
        return (3 >= 3 && 3 <= ( 3 + 9)) || (((3 + 9) >= 3 && (3 + 9) <= (3 + 9)));
```

karena tiga tidak lebih besar dar 3, maka percabangan bernilai salah dan memasuki kondisi ke-dua sehingga boxA adalah b1 dan boxB adalah b2. Selanjutnya

dilakukan pengecekan pada dua kondisi, tiga lebih besar sama dengan tiga dan tiga kurang dari sama dengan dua belas, kondisi ini bernilai *true* jika salah kondisi bernilai *true* maka dinyatakan bahwa dua karakter ini berada dalam satu baris.

Berdasarkan asumsi bahwa semua baris teks dimulai dari kiri ke sebelah kanan, semua kandidat karakter akan disortir terlebih dahulu untuk mendapatkan koordinat horizontal mereka. Kandidat karakter didapatkan dari hasil segmentasi yang dilakukan sebelumnya. Berikut adalah proses untuk perancangan *min-cost flow network.*

1. Pertama, sepasang node diciptakan untuk setiap kandidat karakter dengan dengan ujung diantaranya yang merepresentasikan biaya dari setiap node (*data cost*) pada persamaan (2.4).
2. Kedua, arah diciptakan dari karakter kandidat A ke karakter kandidat B menggunakan *smoothess cost* pada persamaan (2.5).
3. Ketiga, node sumber (S) dan node ujung (T) akan diciptkakan sebagai titik awal dan akhir dari *min-cost flow* ini dan setiap karakter kandidat akan terkoneksi kepada keduanya.



Gambar 3.14 Ilustrasi Jarak dan Persamaan Ukuran Calon Kandidat Karakter

Untuk setiap kandidat karakter A, kandidat karakter berikutnya B, yang dapat disambungkan oleh A, harus dibatasi oleh kondisi tertentu untuk mengurangi kesalahan dalam penghitungan jarak antara karakter A dan B. Berikut adalah kondisi yang harus dipenuhi.

Tabel 3.22 Kondisi Jarak Antara Kandidat Karakter

No	Kondisi	Syarat
1.	Jarak Horizontal antara A dan B	$\frac{H(A,B)}{\min(W_{A,}W_b)} < T_H$
2.	Jarak vertikal Antara A dan B	$\frac{V(A,B)}{\min(W_{A,}W_b)} < T_V$
3.	Ukuran dari A dan B	$\frac{\lvert W_A - W_b \rvert}{\min(W_{A,}W_b)} < T_S$

Setelah dilakukan proses *segmentasi*, maka akan didapatkan calon kandidat yang akan digunakan untuk menentukan *minimum cost flow*. Berikut adalah contoh calon kandidat yang didapat berdasarkan hasil segmentasi, yaitu A sebagai huruf I dan B sebagai huruf N. Berikut adalah masing-masing calon karakter hasil segmentasi.

Berdasarkan Tabel 3.21 diperoleh data sebagai berikut. $W_A = 5$, $W_B = 9$, H(A,B) = 1 dan V(A,B) = 0. Sebelum menentukan nilai atau *cost* untuk memproses *minimmun cost flow*, calon kandidat karakter wajib memenuhi syarat pada tabel Tabel 3.22. Berdasarkan tabel 3.22 maka akan dilakukan pengecekan pada A dan B sebagai berikut.

a. Jarak horizontal antara A dan B untuk $T_H = 2$

   H(A,B) = 1

   $W_A = 5$

   $W_B = 9$

   Maka nilai horizontal antara A dan B adalah

   $$\frac{1}{\min(5,9)} < 2$$

   $$\frac{1}{5} < 2$$

   $$\frac{1}{5} < 2$$

   $$0.2 < 2$$

Berdasarkan pengecekan diatas maka jarak horizontal antara A dan B memenuhi syarat a pada Tabel 3.22.

b. Jarak vertikal antara A dan B untuk $T_v = 0.6$

$V(A,B) = 0$

$W_A = 5$

$W_B = 9$

Maka nilai horizontal antara A dan B adalah

$$\frac{0}{\min(5,9)} < 0.6$$

$$\frac{0}{5} < 0.6$$

$$0 < 0.6$$

Berdasarkan pengecekan diatas maka jarak vertikal antara A dan B memenuhi syarat b pada Tabel 3.22.

c. Ukuran antara A dan B untuk $T_s = 1$

$W_A = 5$

$W_B = 9$

Maka nilai horizontal antara A dan B adalah

$$\frac{|5 - 9|}{\min(5,9)} < 1$$

$$\frac{4}{5} < 1$$

$$0.8 < 1$$

Berdasarkan pengecekan diatas maka ukuran antara A dan B memenuhi syarat c pada Tabel 3.22.

Setelah semua pengecekan dilakukan, diperoleh hasil bahwa semua kondisi terpenuhi dan dapat diartikan bahwa dua karakter A dan B tersebut adalah tetangga dan dapat diartikan sebuah teks.

3.5 Spesifikasi Kebutuhan Perangkat Lunak

Setelah melakukan analisis, spesifikasi kebutuhan perangkat lunak dari aplikasi yang akan dibangun dijelaskan dalam tabel 3.23 berikut:

Tabel 3.23 Spesifikasi Kebutuhan Perangkat Lunak

Nomor	Keterangan
Fungsional	
SKPL-F-01	Sistem dapat menerima masukan berupa citra
SKPL-F-02	Sistem dapat melakukan proses *preprocessing* yaitu *grayscale*, *smoothing* dan *grayscale extension.*
SKPL-F-03	Sistem dapat melakukan proses *training*
SKPL-F-04	Sistem dapat melakukan proses *testing*
Non Fungsional	
SKPL-NF-01	Sistem menggunakan aplikasi berbasis website
SKPL-NF-02	Sistem menggunakan Open CV, Phyton dan Visual Studio 2017 untuk membangun aplikasi
SKPL-NF-03	Sistem menggunakan MySQL untuk membangun database

3.6 Analisis Kebutuhan Non Fungsional

Analisis Kebutuhan Non Fungsional merupakan analisis yang dibutuhkan untuk menentukan spesifikasi kebutuhan sistem. Spesifikasi ini juga meliputi elemen atau komponen-komponen apa saja yang dibutuhkan untuk sistem yang akan di bangun sampai dengan sistem tersebut diimplementasikan. Analisis kebutuhan ini juga menentukan spesifikasi masukan yang diperlukan dalam sistem, keluaran yang akan dihasilkan sistem dan proses yang dibutuhkan untuk mengolah masukan sehingga menghasilkan suatu keluaran yang diinginkan.

Pada analisis kebutuhan sistem non fungsional ini juga dijelaskan analisis mengenai perangkat keras (*hardware*), perangkat lunak (*software*) dan pengguna (*user*) sebagai bahan analisis yang dibutuhkan dalam perancangan sistem yang akan diterapkan.

3.6.1 Kebutuhan Perangkat Keras

Agar Aplikasi ini dapat berjalan dengan baik, maka dibutuhkan perangkat keras yang sesuai dengan kebutuhan aplikasi. Berikut ini adalah spesifikasi minimum perangkat keras yang dibutuhkan agar dapat menjalankan aplikasi ini secara optimal.

a. CPU : Intel Pentium Core 2 Duo @ 2.2 GHz, AMD Athlon XP 1500+
b. RAM : 2 GB
c. VGA : 512 MB
d. Harddisk : 10 GB
e. Peralatan Input : Keyboard dan Mouse
f. Peralatan Output : Monitor 14 Inch

3.6.2 Kebutuhan Perangkat Lunak

Perangkat lunak adalah bagian dari komputer yang memberikan intruksi kepada perangkat keras untuk menjalankan intruksi agar dapat menjalankan suatu sistem didalamnya. Adapun perangkat lunak yang digunakan adalah sebagai berikut:

a. Sistem Operasi : Windows 7
b. Sistem Manajemen Basis Data : MySQL
c. Server Basis Data : XAMPP
d. Pengembangan Aplikasi : Java

3.6.3 Analisis Pengguna

Pengguna untuk perangkat lunak ini adalah seseorang yang dapat mengoperasikan komputer dan mengerti alur kerja daripada aplikasi tersebut.

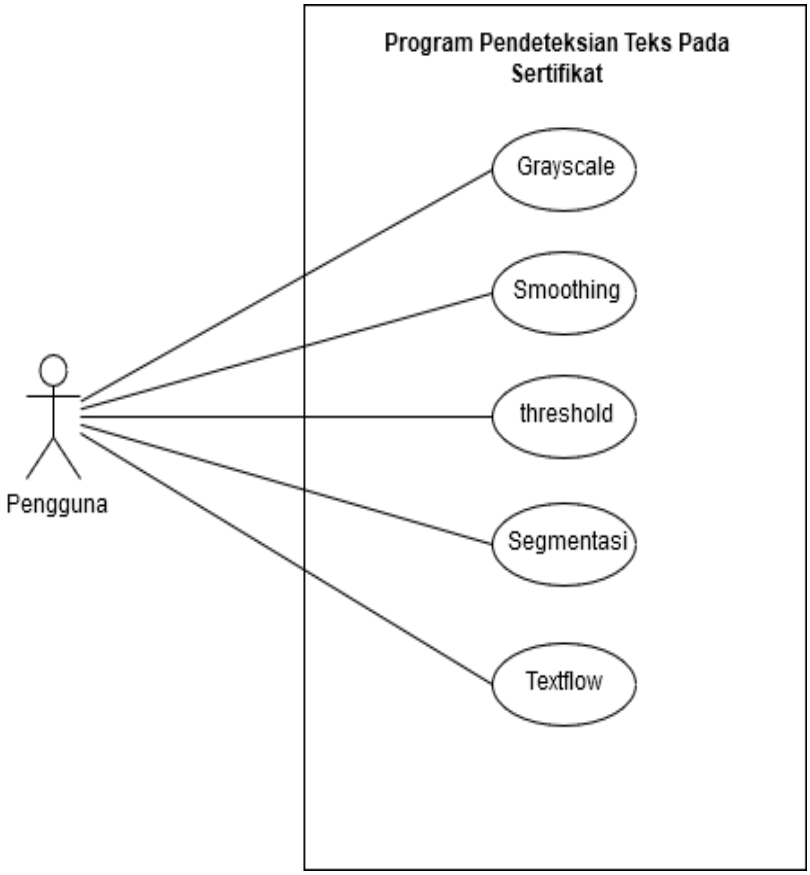
3.7 Analisis Kebutuhan Fungsional

Pemodelan sistem adalah bagian yang menjelaskan fungsi-fungsi yang dapat dilakukan oleh aplikasi berdasarkan analisis metode yang telah dilakukan sebelumnya dan mendeskripsikannya ke dalam bentuk diagram diantaranya

usecase diagram, *diagram scenario*, *activity diagram*, *class diagram*, dan *sequence diagram*.

3.7.1 Usecase Diagram

Usecase Diagram berisi mengenai interaksi antara sekelompok proses dengan sekelompok aktor, menggambarkan fungsionalitas dari sebutah sistem yang dibangun dan bagaimana sistem berinteraksi dengan dunia luar. Usecase diagram dapat digunakan selama proses analisis untuk menangkap kebutuhan sistem dan untuk memahami bagaimana sistem seharusnya bekerja. Berikut Usecase diagram aplikasi pada gambar 3.15.



Gambar 3.15 Use Case Diagram

3.7.2 *Use Case Scenario*

Pada bagian skenario diagram diisi dengan skenario untuk beberapa *usecase* utama pada gambar 3.24 yang menggambarkan urutan interaksi aktor dengan usecase dari awal sampai akhir.

Tabel 3.24 *Use Case Scenario Grayscale*

Nama Usecase:	*Grayscale*	
Tujuan:	Sistem harus mampu mengubah nilai dari citra menjadi derajat keabuan.	
Aktor:	Pengguna	
Kondisi Awal:	Pengguna telah memasukan citra sertifikat kedalam sistem.	
Skenario Utama		
Aksi User		Aksi Sistem
1. Pengguna menekan tombol *Grayscale*		
		2.Sistem menampilkan hasil citra *Grayscale*.
Kondisi Akhir:	Nilai citra berubah menjadi citra *Grayscale*	

Tabel 3.25 *Usecase Scenaro Smoothing*

Nama Usecase:	*Smoothing*	
Tujuan:	Sistem harus mampu mengubah citra menjadi citra smoothing.	
Aktor:	Pengguna	
Kondisi Awal:	Citra Sertifikat telah melalui proses Grayscale	
Skenario Utama		
Aksi User		Aksi Sistem
1. Pengguna menekan tombol *Smoothing*		
		2.Sistem menampilkan hasil citra hasil *Smoothing*.
Kondisi Akhir:	Nilai citra berubah menjadi citra *Smoothing*	

Tabel 3.26 ***Usecase Scenario Threshold***

Nama Usecase:	**Threshold**
Tujuan:	Sistem harus mampu mengubah citra menjadi citra Biner.
Aktor:	Pengguna
Kondisi Awal:	Citra Sertifikat telah melalui proses *Grayscale* dan *Smoothing*
Skenario Utama	
Aksi User	Aksi Sistem
1. Pengguna menekan tombol *Threshold*	
	2.Sistem menampilkan hasil citra hasil *Threshold*.
Kondisi Akhir:	Nilai citra berubah menjadi citra *Threshold / biner*

Tabel 3.27 ***Usecase Scenario*** **Segmentasi**

Nama Usecase:	Segmentasi
Tujuan:	Sistem harus mampu menampilkan bagian huruf pada sertifikat
Aktor:	Pengguna
Kondisi Awal:	Citra Sertifikat telah melalui proses *Grayscale, Smoothing dan Thresholding*
Skenario Utama	
Aksi User	Aksi Sistem
1. Pengguna menekan tombol Segmentasi	
	2.Sistem menampilkan hasil citra hasil Segmentasi dan menampilkan bagian huruf pada serifikat

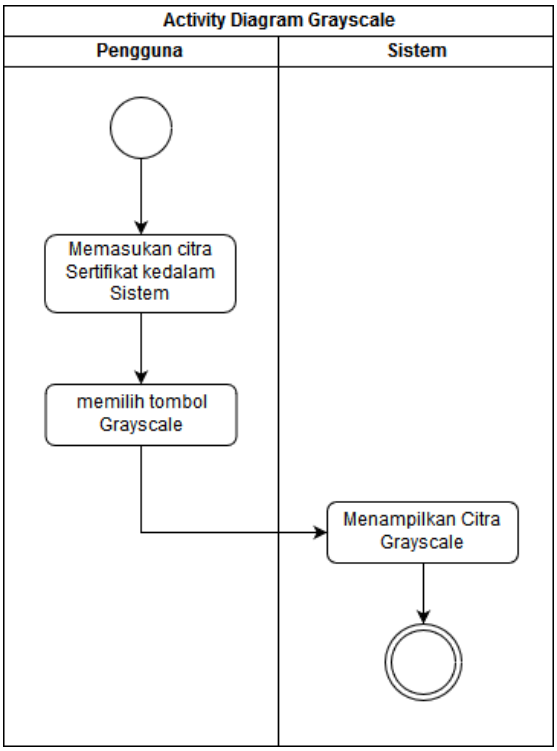
Kondisi Akhir:	Menampilkan bagian bagian huruf pada sertifikat dengan memberi tanda kotak pada huruf yang terdeteksi.

Tabel 3.28 *Usecase Scenario Textflow*

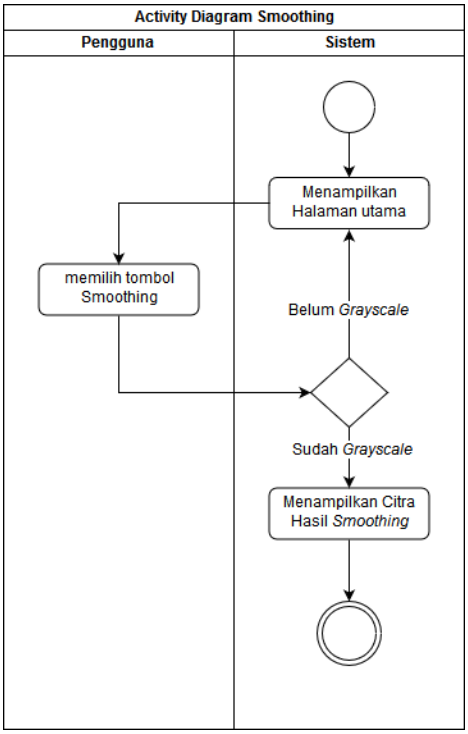
Nama Usecase:	**Textflow**	
Tujuan:	Sistem harus mampu menampilkan bagian teks pada sertifikat	
Aktor:	Pengguna	
Kondisi Awal:	Citra Sertifikat telah melalui proses *Grayscale, Smoothing ,Thresholding* dan Segmentasi	
Skenario Utama		
Aksi User		Aksi Sistem
1. Pengguna menekan tombol *Text Flow*		
		2.Sistem menampilkan hasil citra hasil *Text flow dan menampilkan bagian teks pada serifikat*
Kondisi Akhir:	Menampilkan bagian bagian teks pada sertifikat dengan memberi tanda kotak pada teks yang terdeteksi.	

3.7.3 Activity Diagram

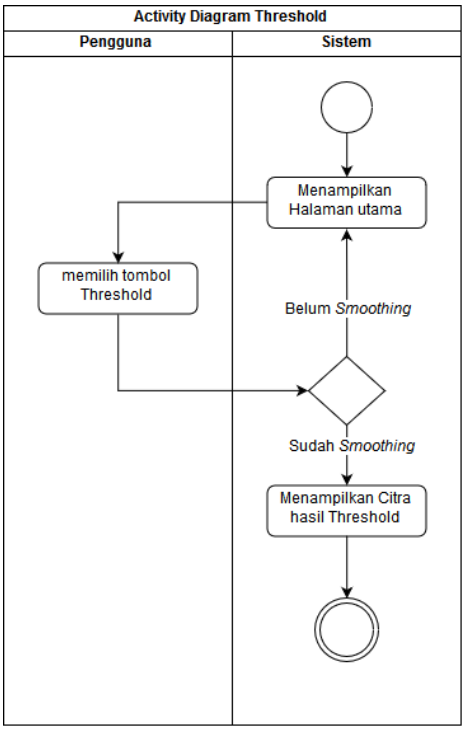
Activity Diagram atau diagram aktivitas menggambarkan berbagai alir aktivitas dalam sistem yang sedang di rancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. Berikut ini merupakan *Activity Diagram* dari aplikasi yang akan di bangun.



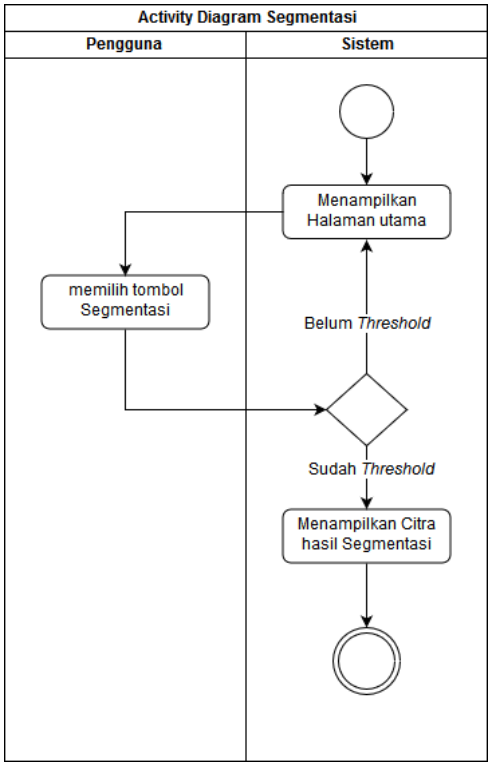
Gambar 3.16 *Acivity* Diagram *Grayscale*



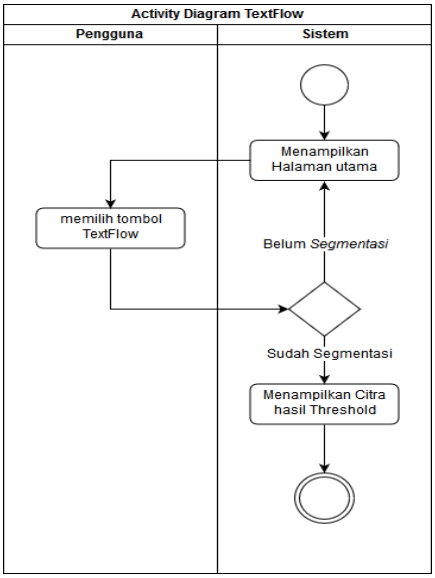
Gambar 3.17 *Acivity* Diagram *Smoothing*



Gambar 3.18 *Activity* Diagram *Threshold*



Gambar 3.19 *Activity* Diagram Segmentasi



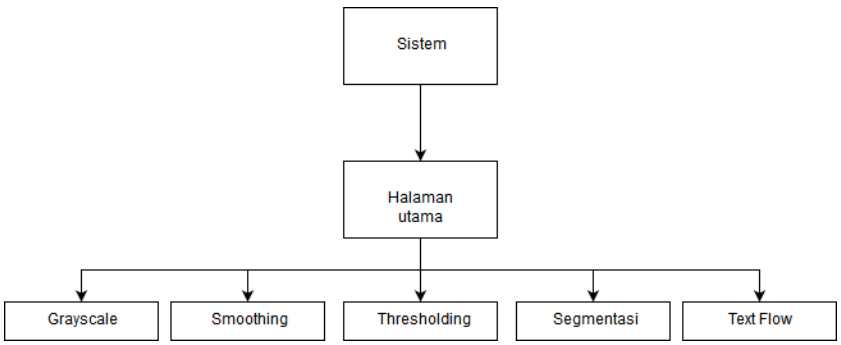
Gambar 3.20 *Activity* Diagram *Textflow*

3.8 Perancangan Sistem

Perancangan sistem merupakan rancangan-rancangan mengenai sistem yang akan di bangun yang terdiri dari rancangan menu, rancangan antarmuka, rancangan antarmuka pesan dan jaringan semantik. Berikut ini merupakan perancangan sistem pada aplikasi yang akan dibangun.

3.8.1 Perancangan Arsitektur Menu

Sebelum membuat perancangan antarmuka, maka dibuat dahulu struktur menu yang terdapat pada program tersebut. Pada gambar 3.21 merupakan perancangan struktur menu pengguna pada sistem ini.

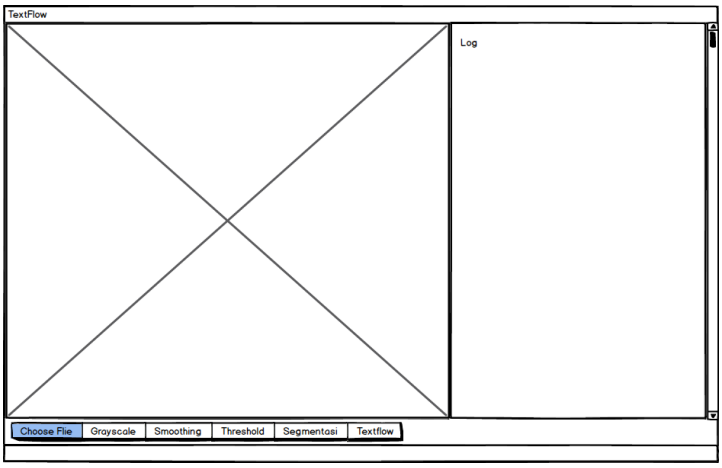


Gambar 3.21 Perancangan Arsitektur Menu

3.8.2 Perancangan Antarmuka

Antarmuka (*Interface*) merupakan mekanisme komunikasi antara pengguna (*user*) dengan sistem yang bertujuan untuk mengkomunikasikan fitur-fitur sistem yang tersedia agar user mengerti dan dapat menggunakan sistem tersebut.

Antarmuka (*Interface*) dapat menerima informasi dari pengguna (*user*) dan memberikan informasi kepada pengguna (*user*) untuk membantu mengarahkan alur penelusuran masalah sampai ditemukan suatu solusi.

F01 – Menu Utama	
	1. Klik tombol “Choose Flie” untuk memasukan gambar yang akan diolah 2. Klik Tombol “Grayscale” untuk melakukan proses Grayscale. 3. Klik tombol “Smoothing” untuk melakukan proses Smoothing. 4. Klik Tombol “Threshold “ untuk

	melakukan proses Threshold. 5. Klik tombol “segmentasi” untuk melakukan proses segmentasi. 6. Klik tombol “textflow “untuk melakukan proses pemilahan kata.

Gambar 3.22 Perancangan Menu Utama