BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Masalah

Analisis masalah dilakukan untuk mengetahui masalah-masalah apa saja yang terjadi dalam pembangunan aplikasi pendeteksian objek pada sebuah citra digital. Pendeteksian objek dalam sebuah citra *(image processing)* adalah salah satu manfaat dari pengolah citra digital. Suatu proses yang digunakan untuk menentukan keberadaan objek tertentu di dalam citra. Proses deteksi tersebut dapat dilakukan dengan berbagai macam metode yang umumnya melakukan pembacaan fitur-fitur dari objek pada citra *input.* fitur dari objek pada citra *input* tersebut akan dibandingkan dengan fitur fitur dari objek referensi atau *template*. Dan hasil perbandingantersebut dapat digunakan untuk menentukan apakah suatu objek terdeteksi sebagai *template* yang dimaksud atau tidak.

Masalah yang terjadi ketika menentukan objek tertentu di dalam suatu citra digital. Proses pembacaan dan perbandingan fitur objek mengalami kesulitan pada saat model yang dijadikan *template* tidak sesuai dengan model pada citra target. Hal ini mengakibatkan objek tersebut tidak dapat terdeteksi dan berakibat pada hasil pembacaan *template* yang salah, terutama saat suatu objek yang dijadikan *template* memiliki tingkat kecerahan yang berbeda dengan citra target. Masalah yang lain terjadi ketika *template* yang diberikan tidak sesuai (mengalami perubahan).

Template Matching adalah salah satu teknik dalam pengolahan citra digital yang berfungsi untuk mencocokan tiap-tiap bagian dari suatu citra. Algoritma ini berkerja dengan cara mengevaluasi pola citra yang akan dibandingkan dengan pola citra *template* pada basis data. Tetapi *template matching* memiliki kekurangan salah satunya yaitu terhadap perubahan intensitas cahaya pada citra, hal ini mengakibatkan objek yang dicari dalam citra tersebut

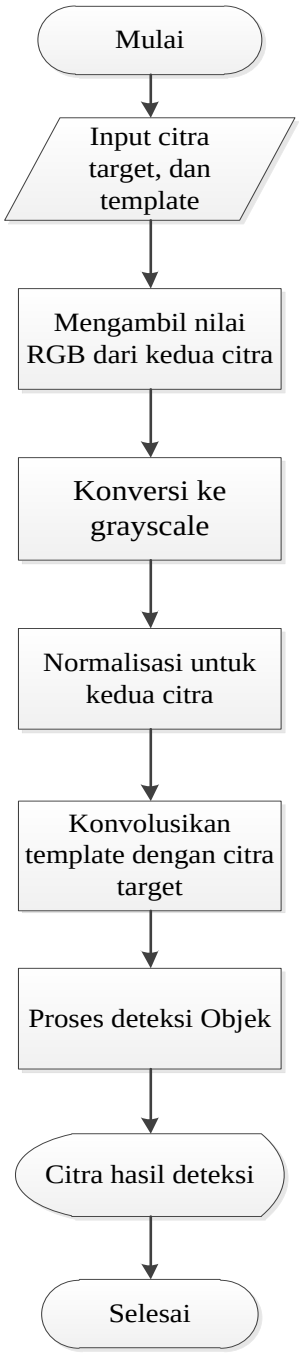
tidak dapat terdeteksi. Dalam penelitian ini akan menganalisis metode yang dapat mengatasi salah satu kekurangan yang terdapat pada *template matching* yaitu dengan menggunakan metode *Normalized Cross Corelation (NCC).* Metode ini sangat cocok digunakan pada *template matching,* fungsi *Normalized Cross Corelation* yang digunakan untuk menormalisasikan intensitas cahaya pada tiap-tiap blok *pixel* dari kedua citra tersebut.

3.2 Analisis Sistem

Analisis sistem merupakan suatu cara yang digunakan dalam mengidentifikasi dan mengevaluasi suatu permasalahan, hambatan yang dihadapi dan kebutuhan yang diharapkan dalam sebuah sistem. Analisa ini diperukan sebagai dasar untuk tahapan perancangan sistem. Hasil dari analisa sistem ini akan menghasilkan berbagai kesimpulan dan saran yang dapat digunakan untuk dijadikan dasar perancangan aplikasi yang akan dibangun. Analisis sistem yang dilakukan untuk membangun aplikasi ini sebagai berikut :

3.2.1 Analisis Sistem yang Akan dibangun

Sistem yang akan dibangun adalah sistem pendeteksian objek pada sebuah citra digital. Sistem ini dapat mencari objek yang kita cari sesuai dengan citra yang menjadi referensi atau *template*. Adapun gambaran umum tentang sistem yang akan dibangun dapat dilihat pada gambar 3.1.



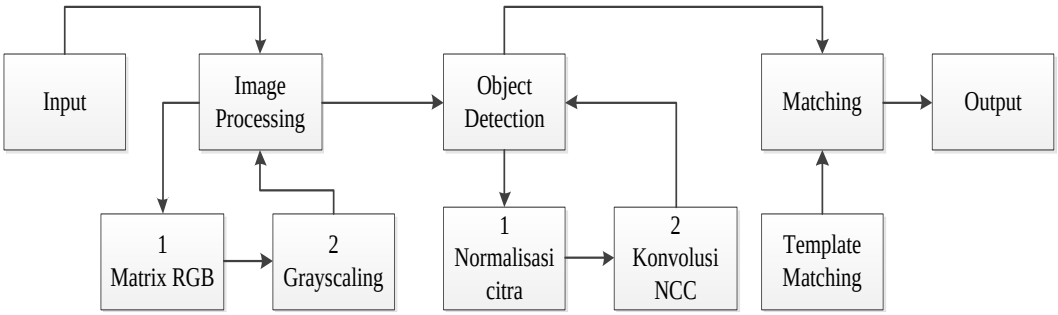
Gambar 3. 1 Gambaran Umum Sistem yang Akan Dibangun

Berikut penjelasan gambaran umum sistem yang akan dibangun pada gambar 3.1 :

1. Pengguna mengunggah citra yang akan dijadikan citra uji (citra target) dan citra yang akan dideteksi/ dijadikan *template* (sebagai acuan).
2. Citra berupa citra RGB. Mengambil nilai GRB dari kedua citra tersebut.
3. Suatu suatu citra warna diubah menjadi citra keabuan (*grayscale*). Untuk memperoleh informasi intensitas dari gambar, sehingga dapat di sortir mulai dari hitam untuk intensitas yang paling lemah sampai dengan putih intensitas yang paling kuat, dan kemudian mendapatkan nilai *grayscale* nya dari masing-masing citra tersebut.
4. Normalisasi, pada tahapan ini kedua citra kemudian di normalisasikan menggunakan tahapan normalisasi yang sesuai dengan metode *Normalized Cross Corelation.*
5. Konvolusi, pada tahapan ini dilakukan proses konvolusi dimana citra dimanipulasi dengan menggunakan eksternal mask / windows untuk menghasilkan citra yang baru. Pada tahapan ini metode *Normalized Cross Corelation* dilakukan.
6. Proses *Matching*, setelah mendapatkan nilai citra yang baru dari konvolusi matriks tahap selanjutnya ialah proses *matching*. Yaitu proses pencocokan atau persamaan antara citra uji dan dengan citra *template.*
7. Outpunya berupa citra yang sudah terdeteksi sesuai objek yang dicari.

3.3 Analisis Metode

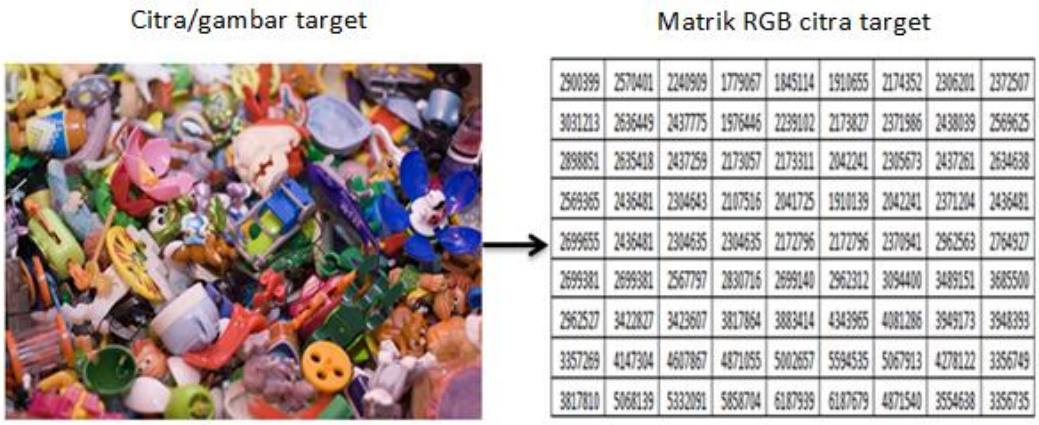
Analisis metode digunakan untuk mengetahui alur proses dari sebuah metode yang digunakan dapat diterapkan ke dalam aplikasi yang dibangun. Pembangunan aplikasi ini menggunakan metode *Normalized Cross Corelation* pada *Template Matching* untuk mencocokan tiap-tiap bagian dari suatu citra dengan citra yang menjadi referensi atau *template*. Adapun penerapan metode pada alur kerja sistem dapat dilihat pada gambar 3.2.



Gambar 3. 2 Penerapan Metode Pada Alur Kerja Sistem

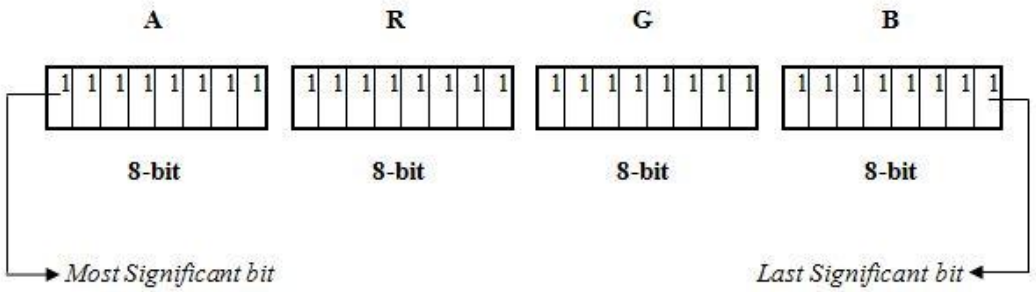
3.3.1 Analisa *RGB Matrix*

RGB merupakan suatu model warna yang terdiri atas 3 buah warna yaitu R(Red), G(Green), dan B(Blue). Pada format RGB, suatu warna didefinisikan sebagai kombinasi (campuran) dari komponen warna R, G dan B. Proses awal yang dilakukan adalah membuatkan matrik RGB dari kedua citra yang akan dianalisis untuk memecah citra menjadi nilai-nilai warna dasar dan mengambil informasi tiga warna dasar dari tiap *pixel* yaitu R,G,B. Citra yang akan dianalisis dapat dilihat pada gambar 3.3.



Gambar 3. 3 Konversi Citra Menjadi Matrik RGB

Setiap warna memiliki 8bit (1byte) dimana memiliki range 256 *color* antara (0-255).



Gambar 3. 4 Formula ARGB 32-bit

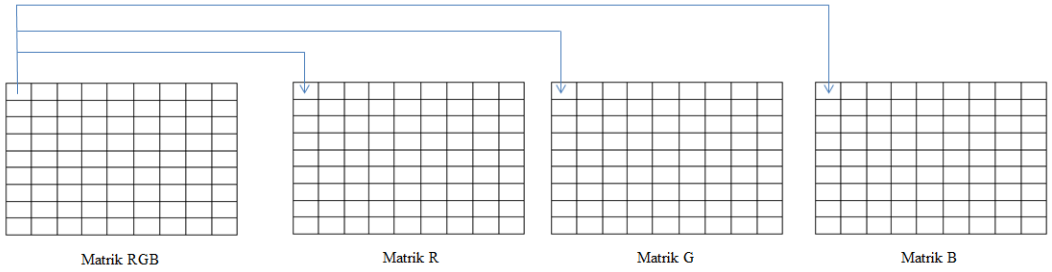
Nilai RGB didapat dengan cara meggeser bit dan di-&kan. Proses pergeseran bit dilakukan dengan cara sebagai berikut :

R = (px >> 16bit) & 255

G = (px>> 8bit) & 255

B = (px & 255)

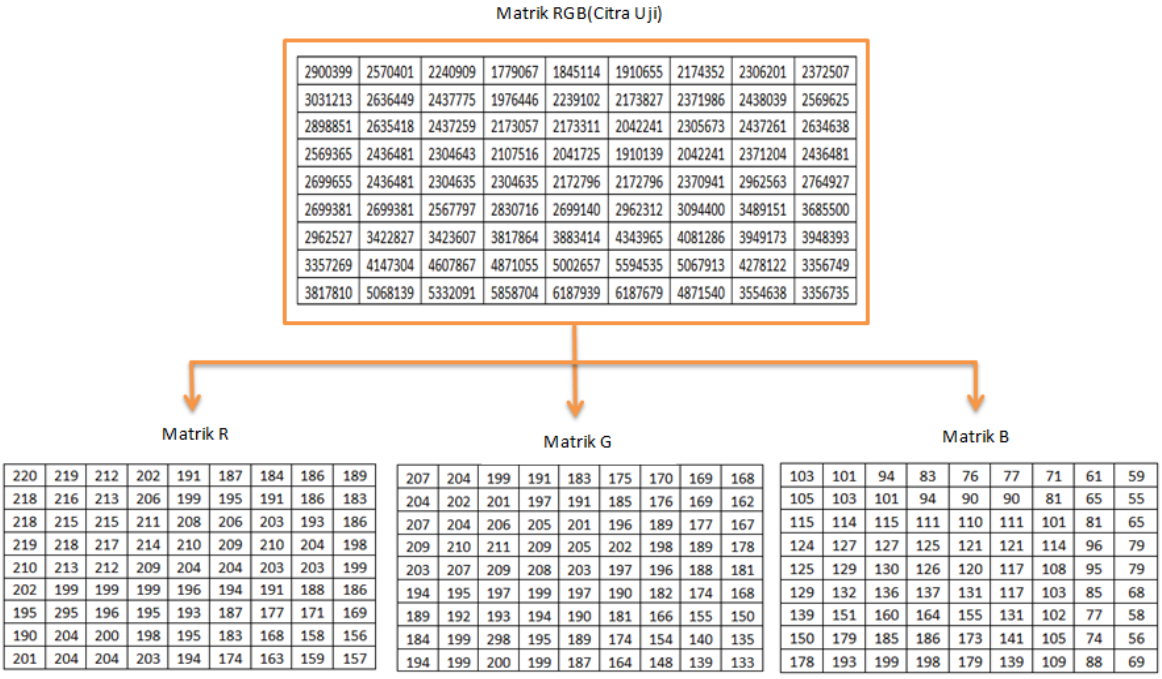
Pengkonversian matrik RGB menjadi masing-masing matrik R, G dan B dapa dilihat pada gambar 3.5, sebagai berikut :



Gambar 3. 5 Konversi Matriks RGB ke Matrik (R, G, B)

Dari Citra matrik RGB di konversi(memecah) citra kedalam masing-masing warna R,G,B untuk diambil 3 warna dasar tiap pikselnya yaitu warna

R(Red), G(Green) dan B(Blue). Dengan ketentuan tinggi dan lebar harus sama antara matrik citra dan aray RGB. Berikut adalah gambar hasil konversi matrik (RGB) ke masing-masing warna R(Red), G(Green), dan B(Blue). Dengan ukuran citra 9 x 9, dapat dilihat pada gambar 3.6.



Gambar 3. 6 Hasil Konversi Matrik RGB Menjadi Matrik (R, G dan B)

Algoritma untuk konversi kedalam masing-masing warna *Red, Green,* dan *Blue* adalah sebagai berikut :

```
(* untuk mendapatkan matrik red, green dan blue dengan cara penggeseran)
     Red =(rgb >> 16) & 255;
     Green =(rgb >> 8) & 255;
     Blue = (rgb & 0xff);
     Alpha =(rgb >> 24) & 255;

Procedure getChanelRGB()

Kamus
   Rgb, height, width : int

Algoritma
        for (int i = 0; i < height; i++)
            for (int j = 0; j < width; j++)
                    CHANNEL_RED:
                       result[i][j] = Red(rgb[i][j]);
                    else
                    CHANNEL_GREEN:
                       result[i][j] = Green(rgb[i][j]);
                   else
                   CHANNEL_BLUE:
                       result[i][j] = getBlue(rgb[i][j]);
            endfor
        endfor
```

3.3.2 Analisis Tahap *Grayscaling*

Proses yang lakukan selanjutnya adalah mengubah citra warna menjadi citra grayscale, hal ini digunakan untuk menyerdanakan model citra. Citra warna terdiri dari 3 layer matrik yaitu R-layer, G-layer, B-layer. Sehingga untuk melakukan proses-proses selanjutnya tetap diperhatikan tiga layer di atas. Bila setiap proses perhitungan dilakukan menggunakan tiga layer, berarti dilakukan tiga perhitungan yang sama. Sehingga konsep itu diubah dengan mengubah 3 layer di atas menjadi 1 layer matrik gray-scale dan hasilnya adalah citra gray-scale. Dalam citra ini tidak ada lagi warna, yang ada adalah derajat keabuan

Untuk mengubah citra berwarna yang mempunyai nilai matrik masing-masing r, g dan b menjadi citra gray scale dengan nilai s, maka konversi dapat dilakukan dengan langkahnya sebagai berikut :

1. Membuat array 2D bertipe integer berukuran *m x n*, dimana (*m*) adalah tinggi citra dan *(n)* adalah lebar citra.
2. Telusuri setiap *pixel (i,j)* pada citra dan hasilkanlah *grayscale* dari *pixel* ke *(i,j)* dengan cara :

$$s = 0.21*R + 0.72*G + 0.07*B$$

dimana :

s = array *(i,j)* dari matrik *grayscale*

r = nilai *red* dari *pixel (i,j)* pada image

g = nilai *green* dari *pixel (i,j)* pada image

b = nilai *blue* dari *pixel (i,j)* pada image

Berikut adalah hasil perhitungan *grayscale* :

diketahui nilai matriks RGB = 220,207,103

$s_{(0,0)}$ = (0.21*220)+(0.72*207)+(0.07*103)

= 46.2 + 149,04 + 7,21

= 202,54

Didapatkan hasil perhitungan untuk $s_{(0,0)}$ = 202,54. Setelah didapatkan seluruh dari hasil perhitungan kemudian dilakukan *image reconstructing* yaitu proses konversi dari matrik ke gambar, maka dari hasil perhitungan *graycale,* dilakukan *rounding* (pembulatan) nilai ke paling dekat. Jadi hasil pembulatan untuk $s_{(0,0)}$ = 202.

Dilakukan perhitungan yang sama untuk seluruh elemen *pixel* pada citra target dan *template,* maka didapatkan citra *grayscale* sebagai berikut :

202	200	194	186	177	171	166	165	165
200	198	197	192	186	180	173	165	159
203	200	202	200	196	192	186	174	164
205	206	206	204	200	198	195	186	175
199	203	204	202	197	193	191	185	178
191	191	193	195	192	186	178	171	165
187	211	191	192	188	179	164	153	148
183	199	270	195	189	174	154	139	134
194	200	201	200	188	164	148	140	134

Gambar 3. 7 Matrik *Grayscale*

Pada pengubahan sebuah gambar menjadi grayscale dapat dilakukan dengan cara mengambil semua *pixel* pada gambar kemudian warna tiap *pixel* akan diambil informasi mengenai 3 warna dasar yaitu merah, biru dan hijau (melalui fungsi warna to RGB), ketiga warna dasar ini akan dijumlahkan menggunakan 0.21*R + 0.72*G + 0.07*B . Dan hasilnya yang akan dipakai untuk memberikan warna pada pixel gambar sehingga warna menjadi *grayscale*, tiga warna dasar dari sebuah pixel akan diset menjadi nilai rata-rata (melalui fungsi RGB to warna). Adapun perubahan citra RGB (citra awal) menjadi citra *grayscale* dapat dilihat sebagai berikut pada gambar 3.8.

Matrik RGB target

2900399	2570401	2240909	1779067	1845114	1910655	2174352	2306201	2372507
3031213	2636449	2437775	1976446	2239102	2173827	2371986	2438039	2569625
2898851	2635418	2437259	2173057	2173311	2042241	2305673	2437261	2634638
2569365	2436481	2304643	2107516	2041725	1910139	2042241	2371204	2436481
2699655	2436481	2304635	2304635	2172796	2172796	2370941	2962563	2764927
2699381	2699381	2567797	2830716	2699140	2962312	3094400	3489151	3685500
2962527	3422827	3423607	3817864	3883414	4343965	4081286	3949173	3948393
3357269	4147304	4607867	4871055	5002657	5594535	5067913	4278122	3356749
3817810	5068139	5332091	5858704	6187939	6187679	4871540	3554638	3356735

Matrik grayscale citra target

202	200	194	186	177	171	166	165	165
200	198	197	192	186	180	173	165	159
203	200	202	200	196	192	186	174	164
205	206	206	204	200	198	195	186	175
199	203	204	202	197	193	191	185	178
191	191	193	195	192	186	178	171	165
187	211	191	192	188	179	164	153	148
183	199	270	195	189	174	154	139	134
194	200	201	200	188	164	148	140	134

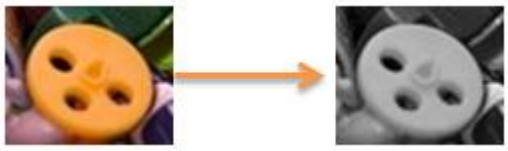
Gambar 3. 8 Matriks RGB (Citra Target) Menjadi *Grayscale*

Berikut ini adalah hasil dari *grayscaling* :



Gambar 3. 9 Hasil *Grayscaling* Untuk Citra Target

Gambar 3.10 adalah hasil *grayscaling* untuk *template :*



Gambar 3. 10 Hasil *grayscaling* Untuk *Template*

Berikut ini adalah matrik hasil dari *grayscaling* :

Matrik RGB template

8436173	9225176	11461368
8699345	9027797	9882336
7185847	9685211	8960975

Matrik grayscale template

81	70	38
77	73	62
101	65	77

Gambar 3. 11 Matrik RGB (*template*) menjadi *Grayscale*

Algoritma untuk proses *grayscale* gambar sebagai berikut :

```
procedure getgrayscale()

kamus
  i,j, width,height : int

Algoritma
  for i <- 0 to width do
     for j <- 0 to height do
     pixelgray=(int)Math.round(0.21*rgbArray[0]+0.71*rgbArray[1]+0.07*rgbArray[2]);
                    gambar[i][j].R <- pixelgray
                    gambar[i][j].G <-pixelgray
                    gambar[i][j].B <-pixelgray
  endfor
endfor
```

3.3.3 Analisis Tahap Normalisasi

Karena metode korelasi yang digunakan adalah *Normalized Cross Corelation* maka perlu adanya normalisasi terlebih dahulu pada window dari

kedua citra yang akan dicari korelasinya, teknik normalisasi yang digunakan sesuai dengan tahapan normalisasi pada *normalized cross corelation.* Berikut adalah tahapan normalisasi untuk kedua citra :

Matrik Citra Target

202	200	194	186	177	171	166	165	165
200	198	197	192	186	180	173	165	159
203	200	202	200	196	192	186	174	164
205	206	206	204	200	198	195	186	175
199	203	204	202	197	193	191	185	178
191	191	193	195	192	186	178	171	165
187	211	191	192	188	179	164	153	148
183	199	270	195	189	174	154	139	134
194	200	201	200	188	164	148	140	134

Matrik template

81	70	38
77	73	62
101	65	77

Gambar 3. 12 Matrik Citra dan Template yang Akan di Normalisasi

1. Tahap menghitung *Mean* (rata-rata) untuk kedua citra.

Diketahui seluruh elemen *pixel* citra uji yang akan di normalisasi sebagai berikut :

g(citra) = (202,200,194,186,177,171,166,165,165,200,198,197,192,186,180,173,165,159,203,200,202,200,196,192,186,174,164,205,206,206,204,200,198,195,186,175,199,203,204,202,197,193,191,185,178,191,191,193,195,192,188,178,171,165,187,211,191,192,188,179,164,153,148,183,199,270,195,189,174,154,139,134,194,200,201,200,188,164,148,140,134)

Setelah di ketahui kumpulan elemen piksel dari sebuah citra tersebut kemudian menghitung rata-rata atau *mean* dari seluruh elemen *pixel* dari citra, perhitungan *mean* ini digunakan untuk menghitung proses normalisasi dari citra. Perhitungan dengan cara membagi jumlah nilai data dengan banyaknya data. Rumus *mean* dapat dituliskan *:*

$$\bar{x} = \frac{x_1 + x_2 + \cdots + x_n}{\text{n}}$$

Maka proses perhitungan sebagai berikut :

g *mean*=

(202+200+194+186+177+171+166+165+165+200+198+197+192+186+180+173
+165+159+203+200+202+200+196+192+186+174+164+205+206+206+204+200
+198+195+186+175+199+203+204+202+197+193+191+185+178+191+191+193
+195+192+188+178+171+165+187+211+191+192+188+179+164+153+148+183
+199+270+195+189+174+154+139+134+194+200+201+200+188+164+148+140
+134)

81

= 185,21.

Elemen *pixel* citra *template* yang akan di normalisasi sebagai berikut :

f(template) = (81,70,38,77,73,62,101,65,77)

Maka proses perhitungan sebagai berikut :

F *mean* = 81+70+38+77+73+62+101+65+77

9

= 71,56

Algoritma untuk proses Normalisasi tuntuk tahap menghitung *mean* untuk kedua citra sebagai berikut :

```
Procedure getMean()

Kamus
        Matrix, height, width : int

Algoritma
        for(int i=0; i<height; i++){
                for(int j=0; j<width; j++){
                        mean +=matrix[i][j];
                        endfor
         endfor
                return mean/(height*width);
endprocedure
```

2. Tahap Menghitung *Standard Deviation (sdv)*

Setelah didapatkan nilai *mean* dari kedua citra tersebut, selanjutnya menghitung standar deviasi untuk kedua citra yang dianalisis untuk mendapatkan nilai citra baru yang sudah dinormalisasikan dengan rumus beriut :

$$\hat{g} = \frac{\mathrm{g} - \bar{g}}{\sqrt{\Sigma(g - \bar{g})^2}}$$

a. Tahap perhitungan *sdv* untuk citra target (g)

Pixel Pada Matrik

px(i,j)									
(0.0)	(0.1)	(0.2)	(0.3)	(0.4)	(0.5)	(0.6)	(0.7)	(0.8)
(1.0)	(1.1)	(1.2)	(1.3)	(1.4)	(1.5)	(1.6)	(1.7)	(1.8)
(2.0)	(2.1)	(2.2)	(2.3)	(2.4)	(2.5)	(2.6)	(2.7)	(2.8)
(3.0)	(3.1)	(3.2)	(3.3)	(3.4)	(3.5)	(3.6)	(3.7)	(3.8)
(4.0)	(4.1)	(4.2)	(4.3)	(4.4)	(4.5)	(4.6)	(4.7)	(4.8)
(5.0)	(5.1)	(5.2)	(5.3)	(5.4)	(5.5)	(5.6)	(5.7)	(5.8)
(6.0)	(6.1)	(6.2)	(6.3)	(6.4)	(6.5)	(6.6)	(6.7)	(6.8)
(7.0)	(7.1)	(7.2)	(7.3)	(7.4)	(7.5)	(7.6)	(7.7)	(7.8)
(8.0)	(8.1)	(8.2)	(8.3)	(8.4)	(8.5)	(8.6)	(8.7)	(8.8)

Matrik citra target

202	200	194	186	177	171	166	165	165
200	198	197	192	186	180	173	165	159
203	200	202	200	196	192	186	174	164
205	206	206	204	200	198	195	186	175
199	203	204	202	197	193	191	185	178
191	191	193	195	192	186	178	171	165
187	211	191	192	188	179	164	153	148
183	199	270	195	189	174	154	139	134
194	200	201	200	188	164	148	140	134

Maka perhitungannya untuk mencari *sdv* untuk citra target(g) sebagai berikut :

Tabel 3. 1 Perhitungan Tahap mencari nilai (sdv) Pada Citra Target

Px (i,j)	Perhitungan (g *mean*=185,21)	Hasil
0,0	$(202-185,21)^2$	289
0,1	$(200-185,21)^2$	255
0,2	$(194-185,21)^2$	81
0,3	$(186-185,21)^2$	1
0,4	$(177-185,21)^2$	64
0,5	$(171-185,21)^2$	196
0,6	$(166-185,21)^2$	361
0,7	$(165-185,21)^2$	400
0,8	$(165-185,21)^2$	400
1,0	$(200-185,21)^2$	255
1,1	$(198-185,21)^2$	169
1,2	$(197-185,21)^2$	144
1,3	$(192-185,21)^2$	49
1,4	$(186-185,21)^2$	1
1,5	$(180-185,21)^2$	25
1,6	$(173-185,21)^2$	144

1,7	$(165\text{-}185{,}21)^2$	400
1,8	$(159\text{-}185{,}21)^2$	676

Dilakukan perhitungan yang sama hingga px(8,8). Setelah itu dilakukan perhitungan selanjutnya sebagai berikut.

$$sdv$$

$$= \sqrt{\frac{\begin{matrix}289+255+81+1+64+196+361+400+400+255+169+144+49\\+1+25+144+400+676+324+255+289+255+121+49+1+121\\49+1+121+411+400+411+361+255+169+100+1+100+196+\\400+4+676+36+49+9+36+441+1024+1369+4+196+7225+\\100+16+121+961+2116+2601+81+255+256+9+441+1369+\\+2025+2601\end{matrix}}{185{,}21}}$$

$$= \sqrt{\frac{34494}{185{,}21}} = \sqrt{186{,}24} = 13{,}64$$

Setelah diketahui nilai sdv dari citra target(g) maka didapatkan matrik baru dari perhitungan sebagai berikut :

Diketahui nilai *g mean* = 185,21 dan *sdv* dari g = 13,64. Maka dilakukan perhitungan dengan rumus $\frac{g-\bar{g}}{\bar{g}}$ dengan hasilnya dibulatkan kepaling dekat. Berikut adalah proses perhitungan normalisasi untuk citra target (g) hingga px(8,8) untuk mendapatkan nilai matrik baru hasil normalisasi.

Tabel 3. 2 Perhitungan Normalisasi Untuk Citra Target

Px (i,j)	Perhitungan $\frac{g-\bar{g}}{\bar{g}}$	Hasil
0,0	(202-185,21)/13,64	1

0,1	(200-185,21)/13,64	1
0,2	(194-185,21)/13,64	0
0,3	(186-185,21)/13,64	0
0,4	(177-185,21)/13,64	0
0,5	(171-185,21)/13,64	-1
0,6	(166-185,21)/13,64	-1
0,7	(165-185,21)/13,64	-1
0,8	(165-185,21)/13,64	-1
1,0	(200-185,21)/13,64	1
1,1	(198-185,21)/13,64	0
1,2	(197-185,21)/13,64	0
1,3	(192-185,21)/13,64	0
1,4	(186-185,21)/13,64	0
1,5	(180-185,21)/13,64	0

Maka hasil normalisasi dari citra target(g) dapat dilihat pada gambar 3.13.

1	1	0	0	0	-1	-1	-1	-1
1	0	0	0	0	0	0	-1	-1
1	1	1	1	0	0	0	0	-1
1	1	1	1	1	0	0	0	0
1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	-1	-1
0	1	0	0	0	0	-1	-2	-2
0	1	6	0	0	0	-2	-3	-3
0	1	1	1	0	-1	-2	-3	-3

Gambar 3. 13 Matrik $\hat{g}$ (Citra Target)

Dilakukan perhitungan yang sama untuk proses normalisasi pada *template* (*f*) dengan rumus sebagai berikut :

$$\hat{f} = \frac{\mathrm{f} - \bar{f}}{\sqrt{\sum(f - \bar{f})^2}}$$

b. Tahap perhitungan sdv untuk *template (f)*

px(i,j)

Pixel pada Matrik		
(0.0)	(0.1)	(0.2)
(1.0)	(1.1)	(1.2)
(2.0)	(2.1)	(2.2)

Matrik *template*

81	70	38
77	73	62
101	65	77

Maka perhitungannya untuk mencari *sdv* untuk *template* (f) sebagai berikut :

Tabel 3. 3 Perhitungan Tahap mencari nilai (sdv) Pada Citra Template

Px (i,j)	Perhitungan (f *mean*=71,56)	Hasil
0,0	$(81\text{-}71{,}56)^2$	89,11

0,1	$(70\text{-}71{,}56)^2$	2,43
0,2	$(38\text{-}71{,}56)^2$	1126
1,0	$(77\text{-}71{,}56)^2$	29,59
1,1	$(73\text{-}71{,}56)^2$	2,07
1,2	$(62\text{-}71{,}56)^2$	91,39
2,0	$(101\text{-}71{,}56)^2$	866,7
2,1	$(65\text{-}71{,}56)^2$	43
2,2	$(77\text{-}71{,}56)^2$	29,59

Setelah itu dilakukan perhitungan selanjutnya sebagai berikut.

$$sdv = \sqrt{\frac{89{,}11 + 2{,}43 + 1126 + 29{,}59 + 2{,}07 + 91{,}39 + 866{,}7 + 43 + 29{,}59}{71{,}56}}$$

$$= \sqrt{\frac{2279{,}88}{71{,}56}} = \sqrt{31{,}86} = 5{,}64$$

Setelah diketahui nilai sdv dari citra target(g) maka didapatkan matrik baru dari perhitungan sebagai berikut :

Diketahui nilai *f mean* = 71,56 dan *sdv* dari f = 5,64. Maka dilakukan perhitungan dengan rumus $\frac{f-\bar{f}}{\hat{f}}$ dengan hasilnya dibulatkan kepaling dekat. Berikut adalah proses perhitungan normalisasi untuk *template* (f) hingga px(2,2) untuk mendapatkan nilai matrik baru hasil normalisasi.

Tabel 3. 4 Perhitungan Akhir dari Normalisasi

Px (i,j)	Perhitungan $\frac{f-\bar{f}}{\hat{f}}$	Hasil
0,0	(81-71,56)/5,64	-11
0,1	(70-71,56)/5,64	-13
0,2	(38-71,56)/5,64	-19
1,0	(77-71,56)/5,64	-12
1,1	(73-71,56)/5,64	-12
1,2	(62-71,56)/5,64	-14
2,0	(101-71,56)/5,64	-7
2,1	(65-71,56)/5,64	-14
2,2	(77-71,56)/5,64	-12

Algoritma untuk proses normalisasi pada tahap menghitung *standard deviation(sdv)* hingga keadaan citra sudah dinormalisasikan, sebagai berikut :

```
Procedure getstandardDeviation()

Kamus
        Matrix, height, width, intensity : int

Algoritma
        double std=0;
        double mean=getMean(matrix, height, width);
        double std=getStandardDeviation(matrix, height, width);
        for(int i=0; i<height; i++){
                for(int j=0; j<width; j++){
                        std  +=Math.pow(matrix[i][j]-mean, 2);
                endfor
         endfor
                return Math.sqrt(std);
                return (intensity-mean)/std;
endprocedure
```

3.3.4 Analisis Tahap Konvolusi

Konvolusi (*convolution*) adalah sebuah proses dimana citra dimanipulasi dengan menggunakan eksternal mask / subwindows untuk menghasilkan citra yang baru. Citra yang dianalisis f(x,y) berukuran 9 x 9 dengan *template* sebagai kernel (mask) 3 x 3.

$\hat{g}(i,j) =$

1	1	0	0	0	-1	-1	-1	-1
1	0	0	0	0	0	0	-1	-1
1	1	1	1	0	0	0	0	-1
1	1	1	1	1	0	0	0	0
1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	-1	-1
0	1	0	0	0	0	-1	-2	-2
0	1	6	0	0	0	-2	-3	-3
0	1	1	1	0	-1	-2	-3	-3

$\hat{f}(i,j) =$

-11	-13	-19
-12	-12	-14
-7	-14	-12

Operasinya dengan cara menempatkan kernel pada sudut kiri atas kemudian hitung nilai pikselnya pada posisi (0,0) dari kernel, dan geser satu piksel ke kanan, kemudian hitung nilai piksel pada posisi (0,0) kernel, begitu

seterusnya hingga geser atau piksel ke bawah, lalu mulai lagi melakukan konvolusi dari sisi kiri citra.

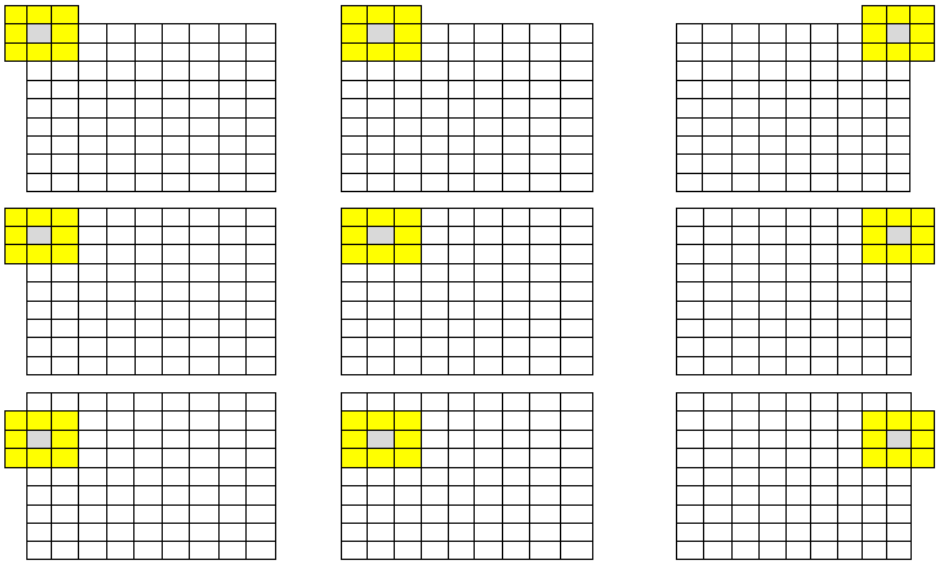
Filter pada citra pada bidang spasial dapat diakukan dengan menggunakan konvolusi (I') dari cita ($\hat{g}$) dan fungsi filternya ($\hat{f}$), dan dituliskan dengan:

$$I' = g \times f$$

Dan dirumuskan dengan :

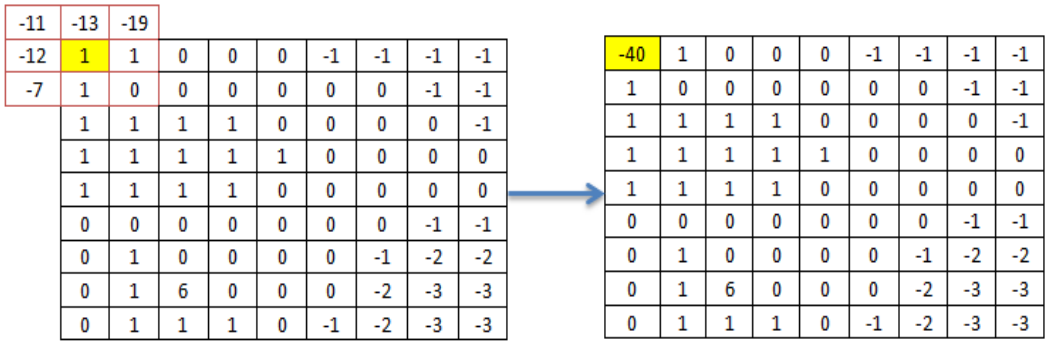
$$I'(\hat{g}, \hat{f}) = \sum_{i=-n}^{n} \sum_{j=-m-1}^{m} h(i,j) I(n,m)$$

Bila dihitung, maka proses perhitungannya dapat dilakukan dengan :



Gambar 3. 14 Perhitungan Konvolusi Secara Grafis

Proses perhitungan konvolusi pada citra yang dianalisis sebagai berikut di mulai dari px(0,0):



-11	-13	-19							
-12	1	1	0	0	0	-1	-1	-1	-1
-7	1	0	0	0	0	0	0	-1	-1
	1	1	1	1	0	0	0	0	-1
	1	1	1	1	1	0	0	0	0
	1	1	1	1	0	0	0	0	0
	0	0	0	0	0	0	0	-1	-1
	0	1	0	0	0	0	-1	-2	-2
	0	1	6	0	0	0	-2	-3	-3
	0	1	1	1	0	-1	-2	-3	-3

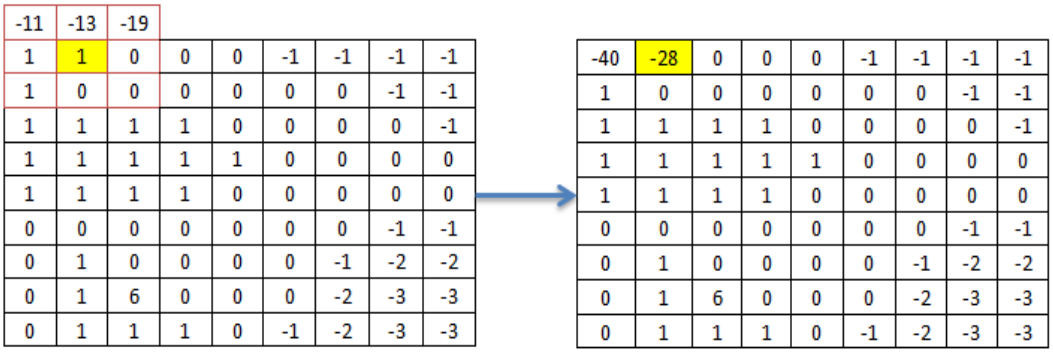
-40	1	0	0	0	-1	-1	-1	-1
1	0	0	0	0	0	0	-1	-1
1	1	1	1	0	0	0	0	-1
1	1	1	1	1	0	0	0	0
1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	-1	-1
0	1	0	0	0	0	-1	-2	-2
0	1	6	0	0	0	-2	-3	-3
0	1	1	1	0	-1	-2	-3	-3

Hasil konvolusi = -40. Nilai ini dihitung dengan cara berikut :

(-11*0)+(-13*0)+(-19*0)+(-12*0)+(-12*1)+(-14*1)+(-7*0)+(-14*1)+(-12*0)

= -40

Dengan cara yang sama, setiap baris piksel dikonvolusi, dengan menggeser 1 piksel ke kanan px(0,1) :



-11	-13	-19							
1	1	0	0	0	-1	-1	-1	-1	
1	0	0	0	0	0	0	-1	-1	
1	1	1	1	0	0	0	0	-1	
1	1	1	1	1	0	0	0	0	
1	1	1	1	0	0	0	0	0	
0	0	0	0	0	0	0	-1	-1	
0	1	0	0	0	0	-1	-2	-2	
0	1	6	0	0	0	-2	-3	-3	
0	1	1	1	0	-1	-2	-3	-3	

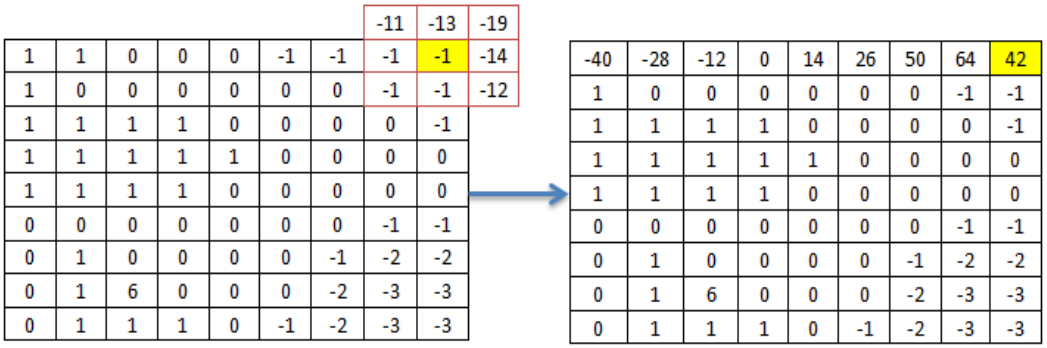
-40	-28	0	0	0	-1	-1	-1	-1
1	0	0	0	0	0	0	-1	-1
1	1	1	1	0	0	0	0	-1
1	1	1	1	1	0	0	0	0
1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	-1	-1
0	1	0	0	0	0	-1	-2	-2
0	1	6	0	0	0	-2	-3	-3
0	1	1	1	0	-1	-2	-3	-3

Hasil konvolusi = -28. Nilai ini dihitung dengan cara berikut :

(-11*0)+(-13*0)+(-19*0)+(-12*1)+(-12*1)+(-14*0)+(-7*1)+(-14*0)+(-12*0)

= -28

Dengan cara yang sama, setiap baris piksel dikonvolusi hingga px(i,j) :



							-11	-13	-19
1	1	0	0	0	-1	-1	-1	-1	-14
1	0	0	0	0	0	0	-1	-1	-12
1	1	1	1	0	0	0	0	-1	
1	1	1	1	1	0	0	0	0	
1	1	1	1	0	0	0	0	0	
0	0	0	0	0	0	0	-1	-1	
0	1	0	0	0	0	-1	-2	-2	
0	1	6	0	0	0	-2	-3	-3	
0	1	1	1	0	-1	-2	-3	-3	

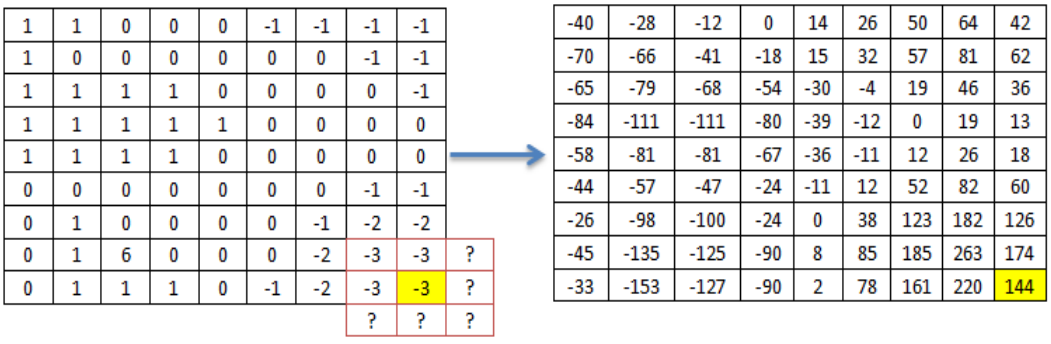
-40	-28	-12	0	14	26	50	64	42
1	0	0	0	0	0	0	-1	-1
1	1	1	1	0	0	0	0	-1
1	1	1	1	1	0	0	0	0
1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	-1	-1
0	1	0	0	0	0	-1	-2	-2
0	1	6	0	0	0	-2	-3	-3
0	1	1	1	0	-1	-2	-3	-3

Hasil konvolusi = 42. Nilai ini dihitung dengan cara berikut :

(-11*0)+(-13*0)+(-19*0)+(-12*(-1))+(-12*(-1))+(-14*0)+(-7*(-1))+(-14*(-1))+(-12*0)

= 42

Dengan cara yang sama, setiap baris piksel dikonvolusi hingga seluruh piksel sudah terkonvolusi.



1	1	0	0	0	-1	-1	-1	-1	
1	0	0	0	0	0	0	-1	-1	
1	1	1	1	0	0	0	0	-1	
1	1	1	1	1	0	0	0	0	
1	1	1	1	0	0	0	0	0	
0	0	0	0	0	0	0	-1	-1	
0	1	0	0	0	0	-1	-2	-2	
0	1	6	0	0	0	-2	-3	-3	?
0	1	1	1	0	-1	-2	-3	-3	?
							?	?	?

-40	-28	-12	0	14	26	50	64	42
-70	-66	-41	-18	15	32	57	81	62
-65	-79	-68	-54	-30	-4	19	46	36
-84	-111	-111	-80	-39	-12	0	19	13
-58	-81	-81	-67	-36	-11	12	26	18
-44	-57	-47	-24	-11	12	52	82	60
-26	-98	-100	-24	0	38	123	182	126
-45	-135	-125	-90	8	85	185	263	174
-33	-153	-127	-90	2	78	161	220	144

Hasil konvolusi = 144. Nilai ini dihitung dengan cara berikut :

(-11*(-3))+(-13*(-3))+(-19*0)+(-12*(-3))+(-12*(-3))+(-14*0)+(-7*0)+(-14*0)+(-12*0)

= 144

Hasil akhir dari tahap konvolusi dapat dilihat pada gambar 3.15.

-40	-28	-12	0	14	26	50	64	42
-70	-66	-41	-18	15	32	57	81	62
-65	-79	-68	-54	-30	-4	19	46	36
-84	-111	-111	-80	-39	-12	0	19	13
-58	-81	-81	-67	-36	-11	12	26	18
-44	-57	-47	-24	-11	12	52	82	60
-26	-98	-100	-24	0	38	123	182	126
-45	-135	-125	-90	8	85	185	263	174
-33	-153	-127	-90	2	78	161	220	144

Gambar 3. 15 Matrik Baru Hasil Konvolusi

3.3.5 *Matched object*

Matched object adalah hasil akhir atau *output* dari proses yang telah melalui semua proses diatas yang menghasilkan nilai maximum dari semua *pixel* yang baru sebagai *center pixel* (nilai yang paling mendekati dengan *template* yang dicari) yang ada di dalam target. Dimana objek yang dicari akan ditandai dengan garis berwarna merah sebagai hasil pendeteksian berupa citra.

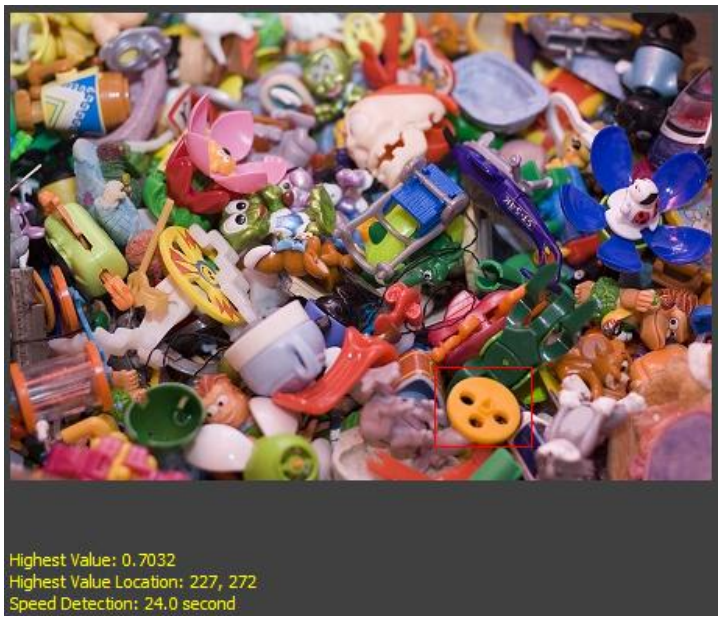
Berikut adalah ilustrasinya :

-40	-28	-12	0	14	26	50	64	42
-70	-66	-41	-18	15	32	57	81	62
-65	-79	-68	-54	-30	-4	19	46	36
-84	-111	-111	-80	-39	-12	0	19	13
-58	-81	-81	-67	-36	-11	12	26	18
-44	-57	-47	-24	-11	12	52	82	60
-26	-98	-100	-24	0	38	123	182	126
-45	-135	-125	-90	8	85	185	263	174
-33	-153	-127	-90	2	78	161	220	144

Gambar 3. 16 Hasil Matrik *Macth*

Setelah diketahui nilai maksimal dari semua elemen *pixel* yang ada dimatrik baru konvolusi, maka *pixel* tersebut adalah *center pixel* dari *template* yang ada dalam citra target.

Adapun output gambar dari hasil deteksi. Dapat dilihat pada gambar 3.17.



Gambar 3. 17 *Output* Gambar Hasil Deteksi

Highest Value(HV) : Didapatkan hasil korelasi(tahap konvolusi) antara *template* dengan citra target dengan hasil 0.7032, dimana nilai tersebut adalah nilai terbesar dari hasil perhitungan korelasi, yang artinya hasil nilai tersebut adalah nilai yang *macth* untuk *template* yang dicari. Sedangkan *Highest Value Location* (HVL) yaitu lokasi *pixel(i,j)* dimana objek yang dicari berada pada titik *pixel* tersebut pada citra target.

Algoritma untuk proses penelusuran citra menggunakan NCC hingga menemukan center *pixel* pada citra target, sebagai berikut :

```
Procedure getNCC()
```

```
Kamus
        MatrixF, matrixG, hightF,hightG, widthF, widthG : int

Algoritma


        for(int i=0; i<heightF; i++){
                for(int j=0; j<widthF; j++){
double normalizedF=getNormalizedIntensity(matrixF[i][j], matrixF, heightF, widthF);
double normalizedG=getNormalizedIntensity(matrixG[i][j], matrixG, heightG, widthG);
          result[i][j]=normalizedF*normalizedG;
          j=widthF;
       endfor
          i=heightF;
    endfor
endprocedure

procedure searchNCC()

Algoritma
    List<Image> values = new ArrayList<Image>();
    for (int i = 0; i < heightTarget - heightTemplate; i++) {
        for (int j = 0; j < widthTarget - widthTemplate; j++) {

                double distance = 0;
                for (int k = 0; k < heightTemplate; k++) {
                        for (int l = 0; l < widthTemplate; l++) {
                   distance += Math.pow(grayTemplate[k][l] - grayTarget[i + k][j + l], 2);
                endfor
                        values.add(new Image(i, j, distance));
        endfor
    endfor
```

```
getHihestIndex()
        List<Cell> cells=new ArrayList<Cell>();
                for(int i=0; i<height; i++){
                for(int j=0; j<width; j++){
                        cells.add(new Cell(i, j, ncc[i][j]));
        endfor
        endfor
endprocedure
```

3.4 Analisis Kebutuhan non-Fungsional

Analisis kebutuhan nonfungsional adalah sebuah langkah dimana seorang pembangun aplikasi menganalisis sumber daya yang dibutuhkan untuk menjalankan aplikasi. Analisis nonfungsional yang dilakukan dibagi menjadi tiga tahap, yaitu analisis pengguna (*user*), analisis kebutuhan perangkat keras, dan analisis perangkat lunak.

3.4.1 Analisis Pengguna

Aplikasi yang dirancang hanya untuk pengguna yang khususnya pengguna yang ingin mengembangkan aplikasi pendeteksi objek pada citra digital, dengan kebutuhan spesifikasi seperti pada tabel 3.5.

Tabel 3. 5 Kebutuhan Pengguna Aplikasi

Spesifikasi	Hak Akses	Tingkat Keterampilan	Jenis Pelatihan
Pengguna	1. Mengupload citra 2. Memilih citra target dan template 3. Melihat hasil dari pendeteksian	Dapat mengoperasikan komputer dan memahami konsep template matching	Tidak diperlukan pelatihan khusus.

3.4.2 Analisis Kebutuhan Perangkat Keras

Perangkat keras merupakan salah satu kebutuhan yang sangat penting bagi penulis dalam membuat aplikasi ini. Peragkat keras akan mempengaruhi kinerja dari pembuatan aplikasi, semakin tinggi spesifikasi dari perangkat keras yang digunakan maka akan semakin cepat pula pembuatan aplikasi tersebut yaitu seperti tabel 3.6.

Tabel 3. 6 Spesifikasi Perangkat Keras

No	Nama Perangkat	Spesifikasi
1	Prosessor	Intel Core i3 2.30 GHz
2	Monitor	Monitor 14inch 1366 x 768 (32bit) (60Hz)
3	Memori	RAM 4 GB DDR3
4	Hardisk	500 GB SATA

Sedangkan kebutuhan perangkat keras untuk menjalankan aplikasi yang dibangun, yang harus dipenuhi yaitu seperti pada tabel 3.7.

Tabel 3. 7 Spesifikasi Perangkat Keras

No	Nama Perangkat	Spesifikasi
1	Prosessor	Intel Core i3 2.30 GHz
2	Monitor	Monitor 14inch 1366 x 768 (32bit) (60Hz)
3	Memori	RAM 1 GB DDR3
4	Hardisk	250 GB SATA

3.4.3 Analisis Perangkat Lunak

Sedangkan Perangkat lunak digunakan dalam sebuah sistem untuk memberi perintah kepada peangkat keras agar bisa saling berinteraksi diantara keduanya. Perangkat lunak yang dibutuhkan dalam membangun dalam membangun aplikasi ini yaitu seperti pada tabel 3.8.

Tabel 3. 8 Spesifikasi Perangkat Lunak

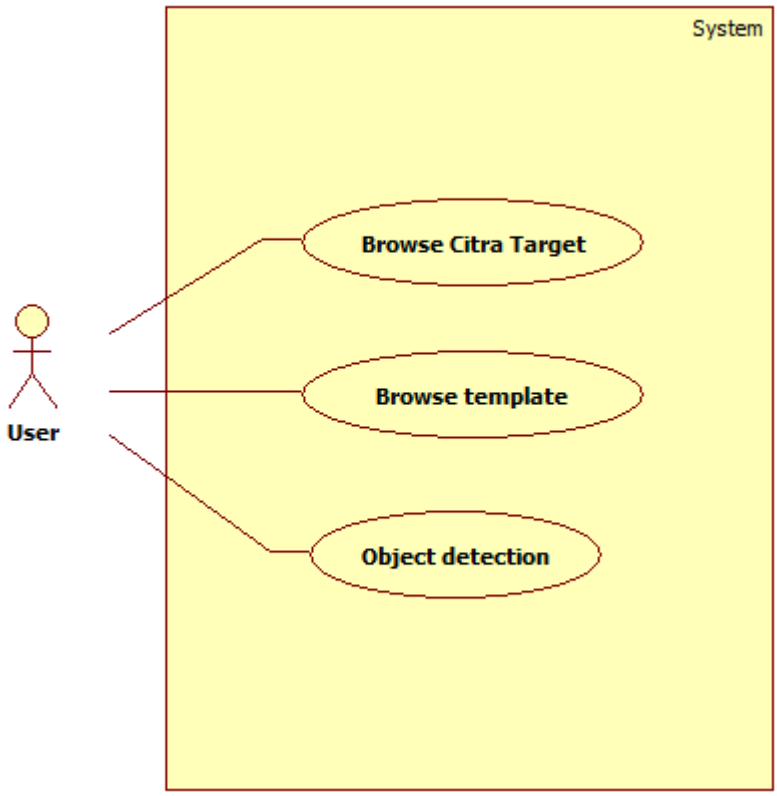
No	Nama Perangkat	Spesifikasi
1	Sistem Operasi	Windows 7 Profesional (64-bit)
2	JDK (*Java Development Kit*)	JDK 1.8.0
3	JRE (*Java Runtime Environment*)	JRE 8.0

3.5 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional adalah segala bentuk data yang dibutuhkan oleh sistem agar sistem dapat berjalan sesuai prosedur dengan prosedur yang dibangun. Aplikasi yang dibangun akan dimodelkan menggunakan *Unified Modeling Language* (UML), dan *tools* yang digunakan yaitu *use case* diagram, *activity* diagram, *sequence* diagram, dan *class* diagram.

3.5.1 *Use Case* Diagram

Use case diagram digunakan untuk mengetahui apa saja yang dapat dilakukan oleh pengguna/ aktor terhadap fungsionalitas yang terdapat pada aplikasi yang dibangun. *Use case diagram* pada aplikasi initerlihat pada gambar 3.18.



Gambar 3. 18 *UseCase Diagram*

3.5.1.1 Identifikasi *Actor*

Actor adalah abstraksi dari orang dan sistem yang lain yang mengaktifkan fungsi dari target sistem. Berikut adalah faktor yang berperan dalam menjalankan aplikasi yang dibangun.

Tabel 3. 9 *Use Case Actor*

No	Aktor	Deskripsi
A-01	User	Merupakan aktor dari aplikasi yang dibangun atau pengguna aplikasi.

3.5.1.2 Identifikasi Use Case Diagram

Berikut tabel identifikasi *Use Case* yang terdapat pada aplikasi :

Tabel 3. 10 *Use Case Actor*

No	Aktor	Deskripsi
UC-01	Browse Target	Merupkan aksi untuk mengambil gambar pada folder galeri.
UC-02	Browse Template	Merupkan aksi untuk mengambil *template* pada folder galeri.
UC-03	Match Process	Merupakan aksi untuk melakukan proses pendeteksian objek.

3.5.2 Skenario *Use Case*

Skenario *use case* merupakan penjelasan secara rinci setiap *use case* yang terdapat pada diagram *use case.* Sekenario *use case* pada aplikasi ini yaitu sebagai berikut :

3.5.2.1 Skenario Pemilihan Citra Uji (Terget)

Skenario *use case* pemilihan citra uji dapat dilihat pada tabel 3.11.

Tabel 3. 11 Skenario *Use Case* Pemilihan Citra

Identifiksi		
Nama *Use Case*	Pemilihan Citra Target	
Aktor	User	
Tujuan	Digunakan untuk memilih *file* citra yang akan digunakan sebagai data yang akan di uji.	
Kondisi Awal	Sistem menampilkan menu pemilihan citra uji	
Skenario Utama		
Aktor		**Reaksi Sistem**
1. User menekan tombol *browse citra*		

	2. Sistem menampilkan dialog *open file*
3. User memilih citra target yang akan digunakan	
4. User menekan tombol *open*	
	5. Sistem menampilkan informasi citra

3.5.2.2 Skenario Pemilihan Citra Template

Skenario *use case* pemilihan *template* dapat dilihat pada tabel 3.12.

Tabel 3. 12 Skenario *Use Case* Pemilihan *Template*

Identifiksi	
Nama *Use Case*	Pemilihan *template*
Aktor	User
Tujuan	Digunakan untuk memilih *file template* yang akan digunakan sebagai referensi (acuan)
Kondisi Awal	Sistem menampilkan menu pemilihan *template*
Skenario Utama	
Aktor	**Reaksi Sistem**
1. User menekan tombol *browse template*	
	2. Sistem menampilkan dialog *open file*
3. User memilih template yang akan digunakan	
4. User menekan tombol *open*	
	5. Sistem menampilkan informasi *template*

3.5.2.3 Skenario Object Detection

Skenario *use case object detection* dapat dilihat pada tabel 3.13.

Tabel 3. 13 Skenario *Use Case Object Detection*

Identifiksi	
Nama *Use Case*	*Object Detection*
Aktor	User
Tujuan	Digunakan untuk melakukan proses pencarian objek yang dipilih
Kondisi Awal	Sistem menampilkan menu *object detection*
Skenario Utama	
Aktor	**Reaksi Sistem**
1. User menekan tombol *match process*	
	2. Sistem menganalisis citra RGB
	3. Sistem mengubah citra RGB ke *Grayscale*
	4. Sistem melakukan proses konvolusi pada citra
	5. Sistem melakukan proses matching objek
	6. Sistem menampilkan hasil pencarian objek

3.5.3 Activity Diagram

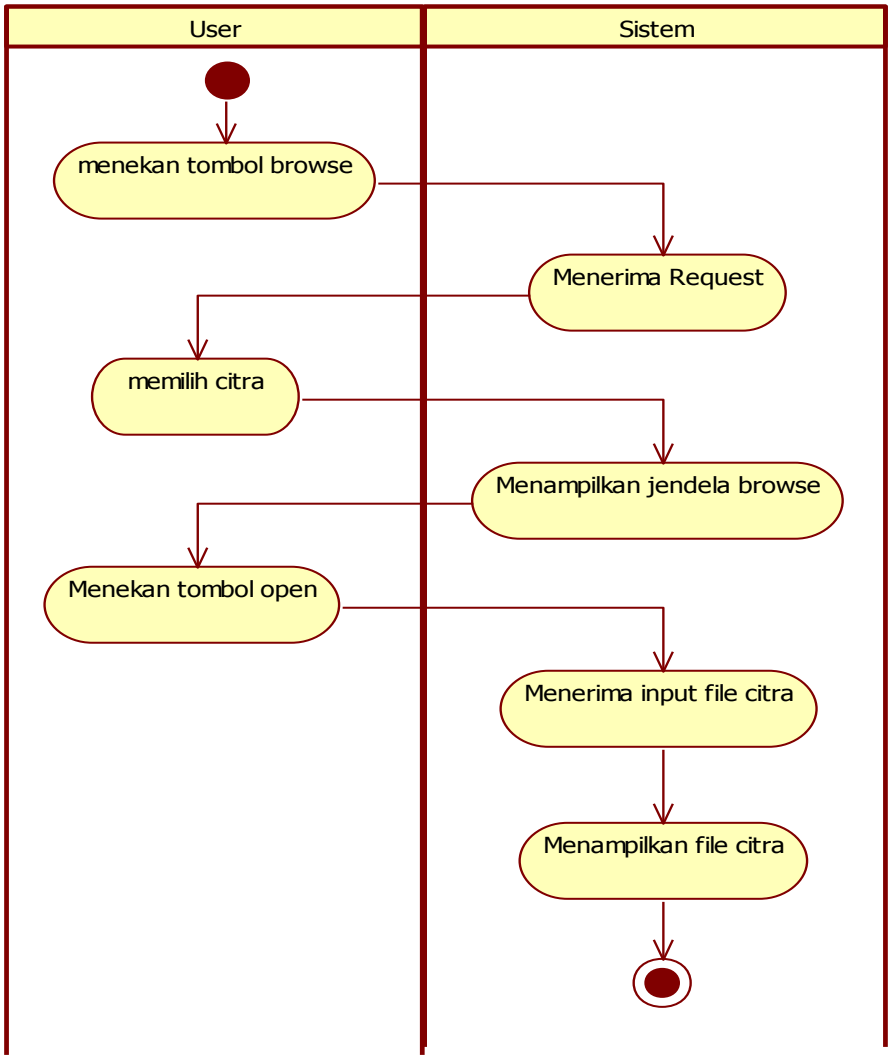
Activity diagram menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram yang terdapat pada aplikasi yang dibangun yaitu sebagai berikut:

3.5.3.1 Activity Diagram Pemilihan Citra Target

Activity diagram pemilihan citra menggambarkan alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alur berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram yang terdapat pada aplikasi yang dibangun yaitu sebagai berikut :

3.5.3.2 *Activity* Diagram Pemilihan Citra Target

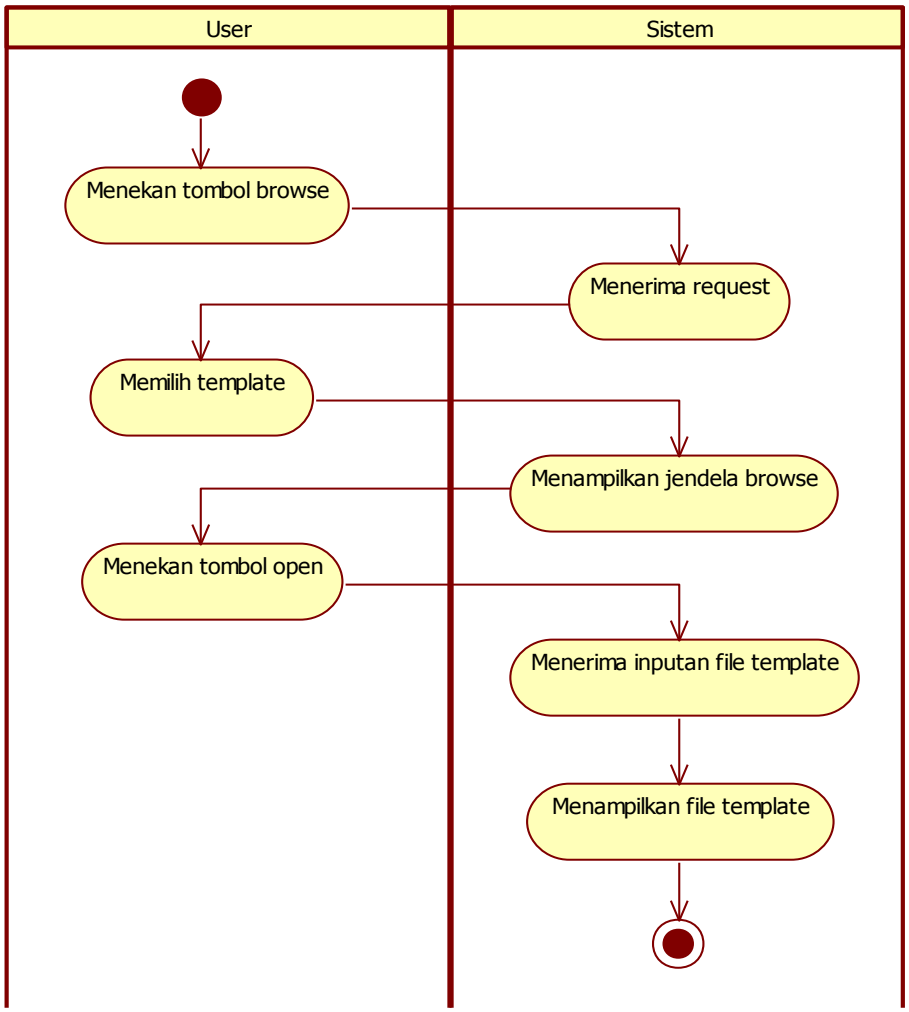
Activity diagram pemilihan citra menggambarkan alur aktivitas pengambilan *file* citra antara pengguna dengan sistem seperti terlihat pada gambar 3.19.



Gambar 3. 19 *Activity* Diagram Pemilihan Citra Target

3.5.3.3 *Activity* Diagram Pemilihan *Template*

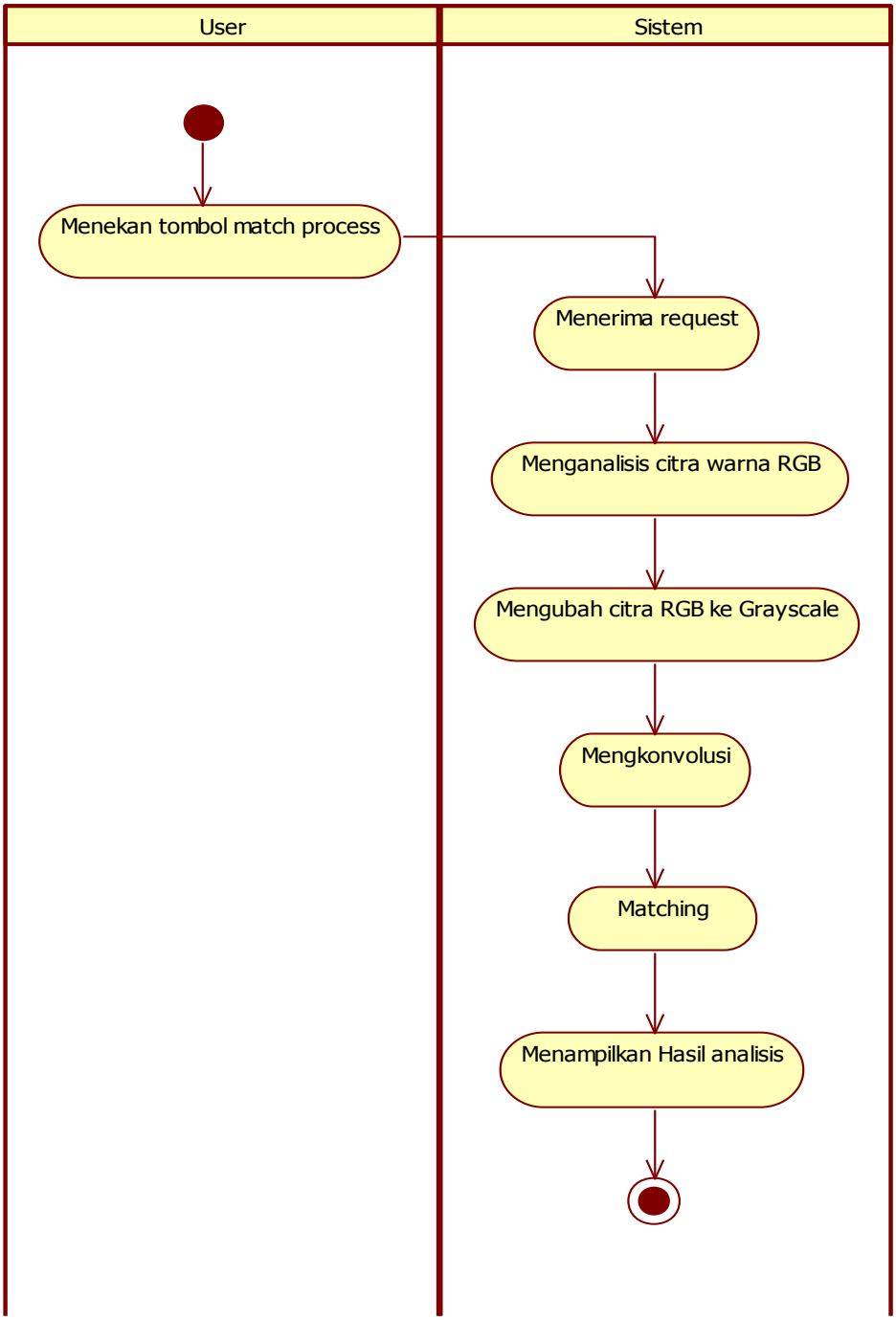
Activity diagram pemilihan *template* menggambarkan alur aktivitas pengambilan *file* citra *template* antara pengguna dengan sistem seperti terlihat pada gambar 3.20.



Gambar 3. 20 *Activity* Diagram Pemilihan *Template*

3.5.3.4 Activity Object Detection

Activity diagram proses pencarian menggambarkan alur aktivitas proses yang dilakukan antara pengguna dengan sistem seperti terlihat pada gambar 3.21.



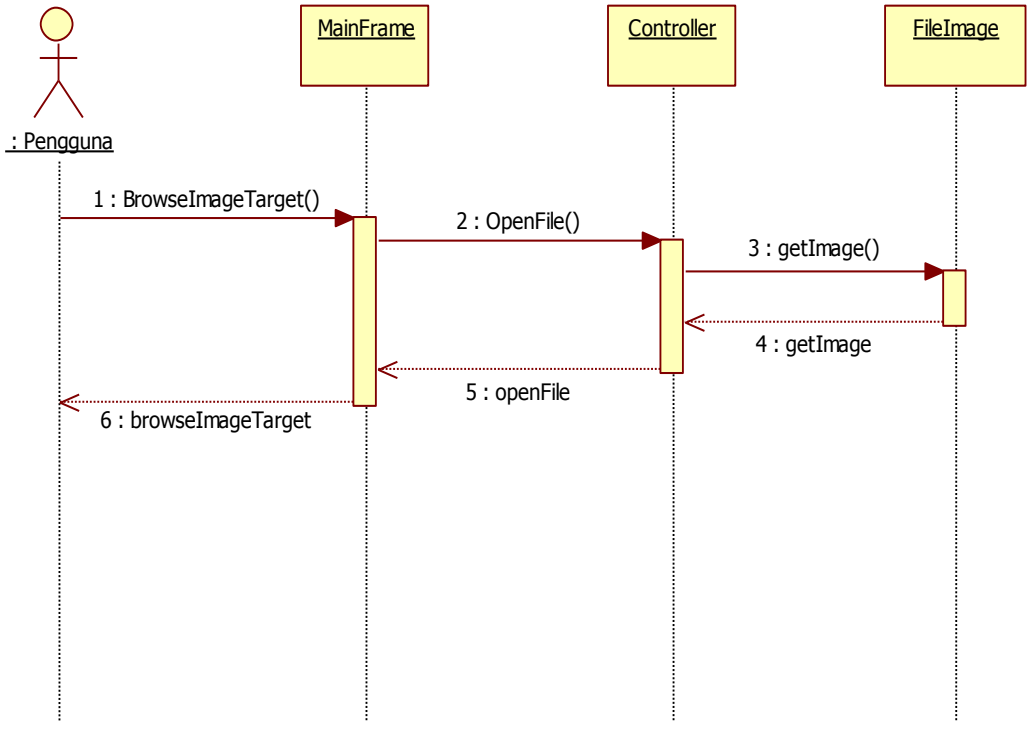
Gambar 3. 21 *Activity* Diagram *Object Detection*

3.5.4 Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem yang digambarkan terhadap waktu. *Sequence* diagram yang terdapat pada aplikasi ini yaitu sebagai berikut :

3.5.4.1 *Sequence* Diagram Pemilihan Citra

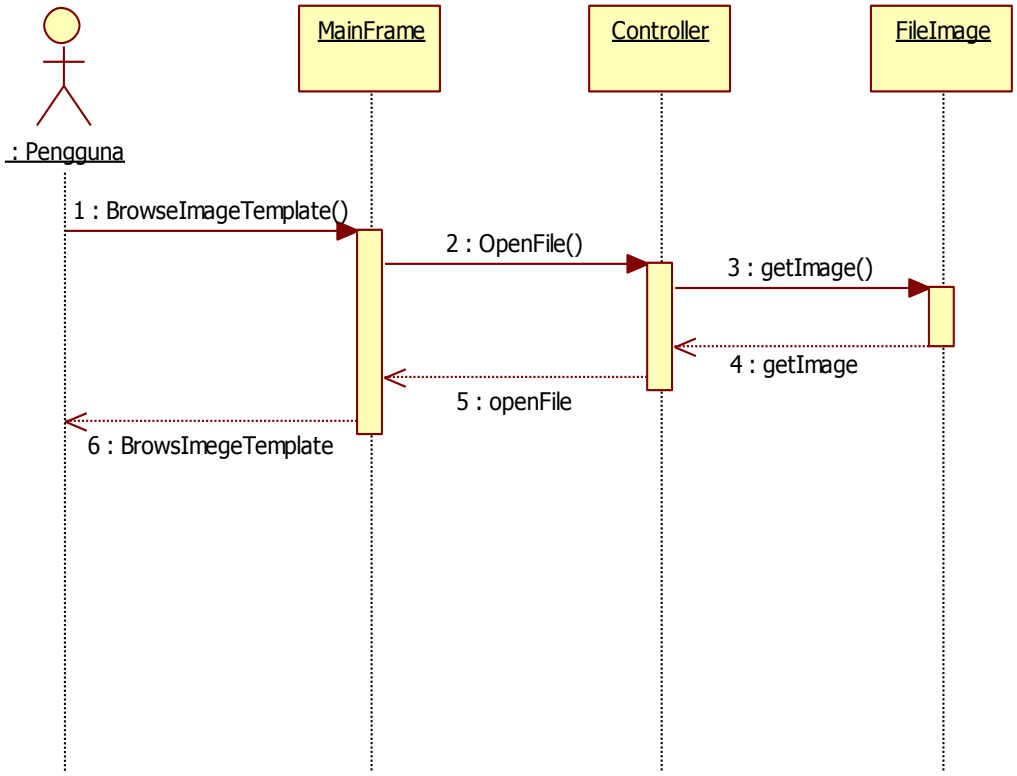
Sequence diagram pemilihan citra merupakan diagram yang menggambarkan interaksi yang terjadi didalam sistem antara pengguna dengan sistem dalam pengambilan *file* citra. *Sequence* diagram pemilihan citra terlihat seperti pada gambar 3.22.



Gambar 3. 22 Sequence Diagram Pemilihan Citra Target

3.5.4.2 *Sequence* Diagram Pemilihan *Template*

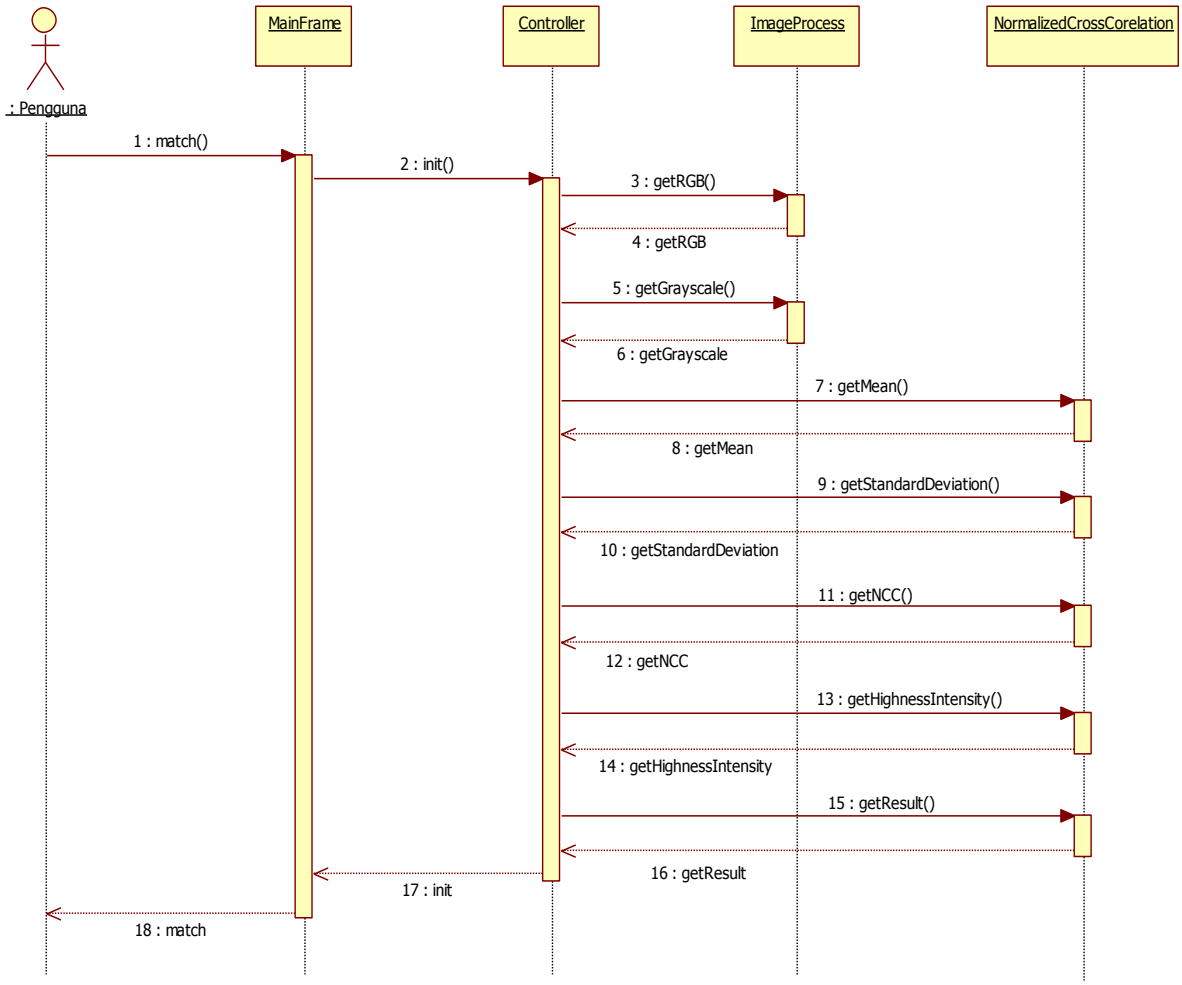
Sequence diagram pemilihan *template* merupakan diagram yang menggambarkan interaksi yang terjadi didalam sistem antara pengguna dengan sistem dalam pengambilan *file template*. *Sequence* diagram pemilihan *template* terlihat seperti pada gambar 3.23.



Gambar 3. 23 *Sequence* Diagram Pemilihan *Template*

3.5.4.3 Sequence Diagram *Object Detection*

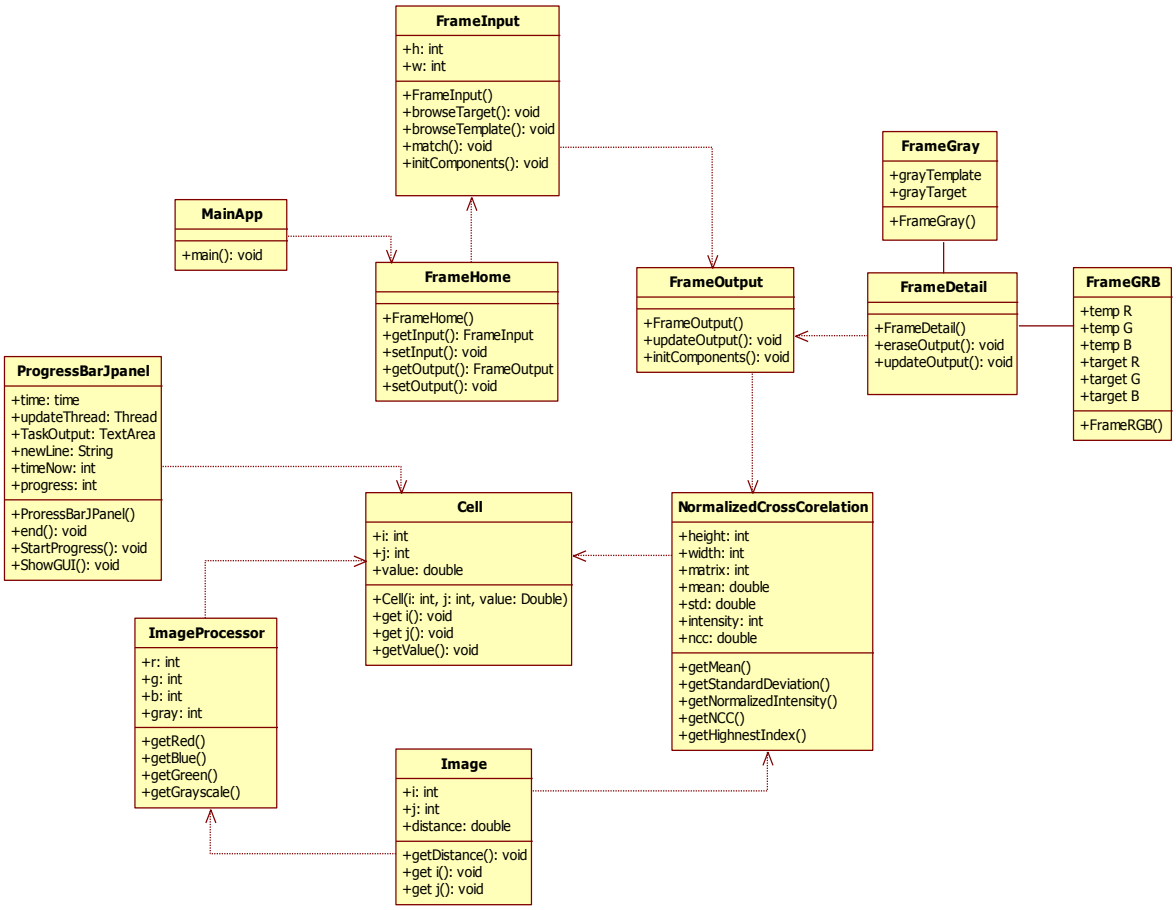
Sequence diagram *object detection* merupakan diagram yang menggambarkan interaksi yang terjadi didalam sistem antara pengguna dengan sistem dalam proses *matching*. *Sequence* diagram *object detection* terlihat seperti pada gambar 3.24.



Gambar 3. 24 *Sequence* Diagram *Object Detection*

3.5.5 Class Diagram

Class diagram digunakan untuk menggambarkan keadaan suatu sistem dengan menjelaskan keterhubungan antara suatu *class* yang lain yang terdapat pada sistem. *Class* diagram yang digunakan pada pembangunan aplikasi pendeteksi objek pada citra digital yaitu pada gambar 3.25.



Gambar 3. 25 Class Diagram Aplikasi Pendeteksi Objek Pada Citra Digital

Tabel 3. 14 Deskripsi *Class Diagam*

Class	Jenis Class	Deskripsi
Main App	Boundary	Merupakan *class* yang digunakan untuk manampilkan Halaman Utama.
FrameHome	View	*Class* yang menampilkan Menu Utama Aplikasi
FrameInput	View	*Class* yang menampilkan *database user*
FrameOutput	View	*Class* yang menampilkan hasil dari deteksi objek
FrameDetail	View	*Class* yang menampilkan detail hasil deteksi
FrameRGB	View	*Class* yang menampilkan hasil dari konversi warna R,G,dan B
FrameGray	View	*Class* yang menampilkan hasil konversi kedalam *grayscale*
Cell	Control	*Class* yang berisi *method-method* untuk pengontrol sistem.
Image	Entity	*Class* yang digunakan untuk menyimpan database *image* dalam sistem.
ImageProcess	Control	*Class* yang berisi *method-method pre-processing image.*

Normalized Cros Corelation	Control	*Class* yang berisi *method-method* dalam tahapan metode NCC
ProgressBarJpanel	Interface	*Class* yang digunakan untuk menampilkan progress bar atau loading.

3.6 Perancangan Sistem

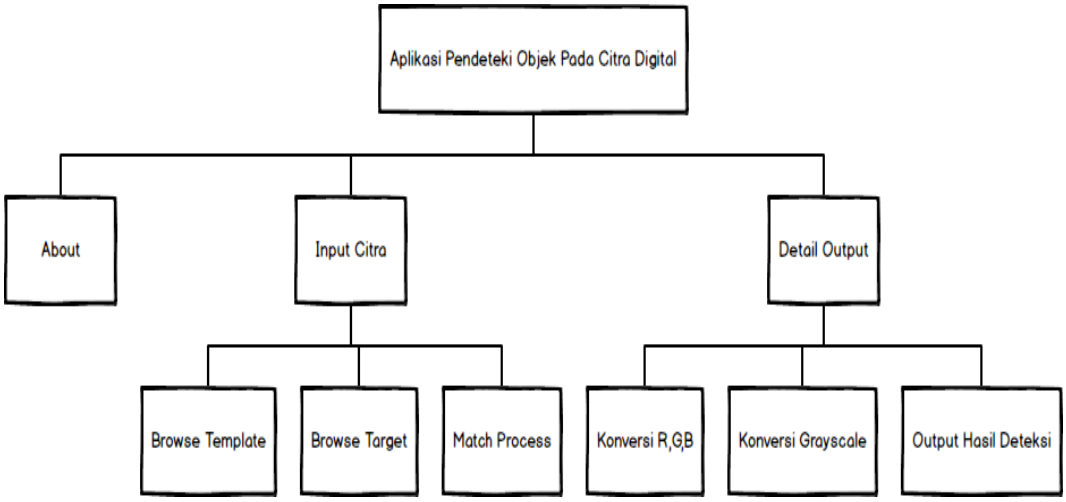
Perancangan sistem merupakan suatu proses yang menggambarkan bagaimana suatu sistem dibangun untuk memenuhi kebutuhan pada fase analisis. Tahap perancangan terdiri dari dua bagian yaitu perancangan arsitektur, dan perancangan *method.*

3.6.1 Perancangan Arsitektur

Perancangan arsitektur terdiri dari perancangan struktur menu, perancangan antarmuka, dan perancangan jaringan sematik.

3.6.1.1 Perancangan Struktur Menu

Perancangan menu dilakukan untuk mempermudah interaksi antara sistem dengan pengguna. Gambar 3.26 merupakan struktur menu dari aplikasi yang akan dibangun.



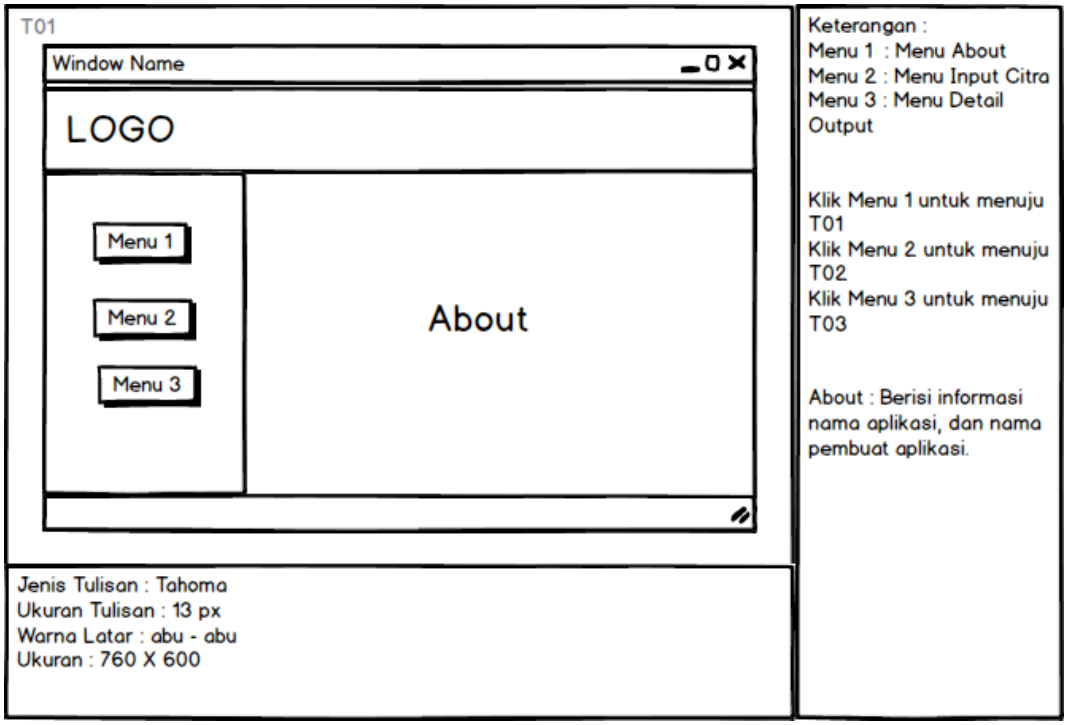
Gambar 3.26 Struktur Menu Aplikasi Pendeteksi Objek Pada Citra Digital

3.6.1.2 Perancangan Antarmuka

Perancangan antarmuka adalah tahapan pembuatan rancangan antarmuka untuk digunakan pada pembangunan aplikasi Pendeteksi Objek Pada Citra Digital yang dibagi menjadi 3 bagian yaitu perancangan antarmuka *about*, perancangan antarmuka *input* citra, perancangan antarmuka *Output* hasil, dan perancangan pesan.

3.6.1.2.1 Perancangan Antarmuka About (T01)

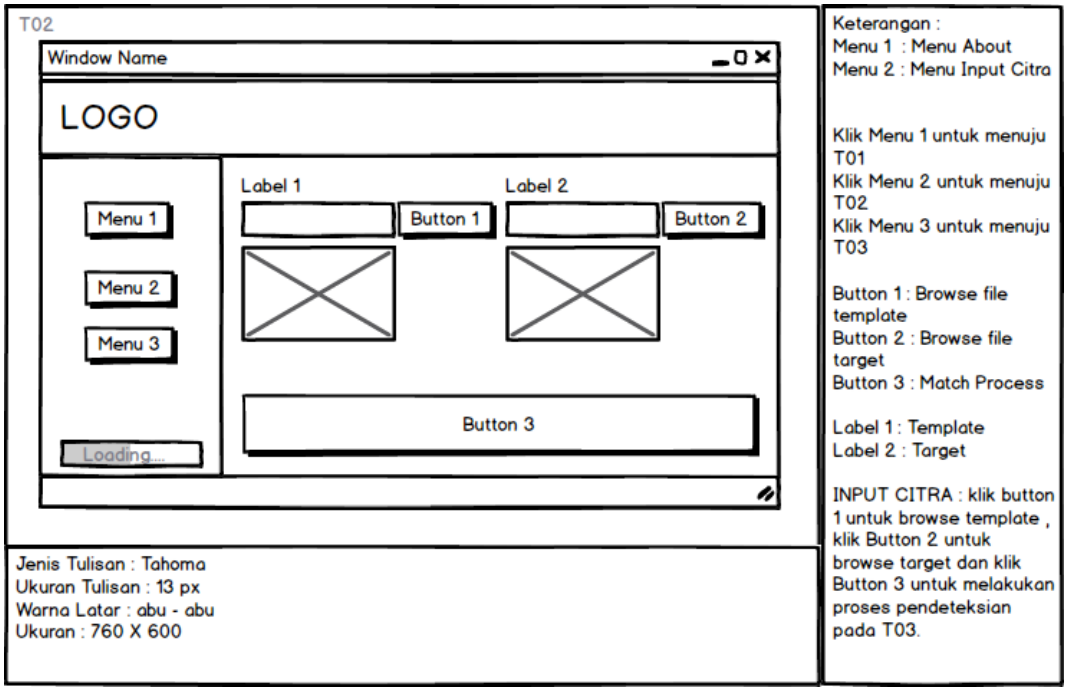
Perancangan antarmuka *About* merupakan tahapan rancangan antarmuka dari awal aplikasi dijalankan. Perancangan antarmuka *home* dari aplikasi pendeteksi objek pada citra digital ini terlihat pada gambar 3.27.



Gambar 3.27 Perancangan Antarmuka *About*

3.6.1.2.2 Perancangan Antarmuka Input Citra (T02)

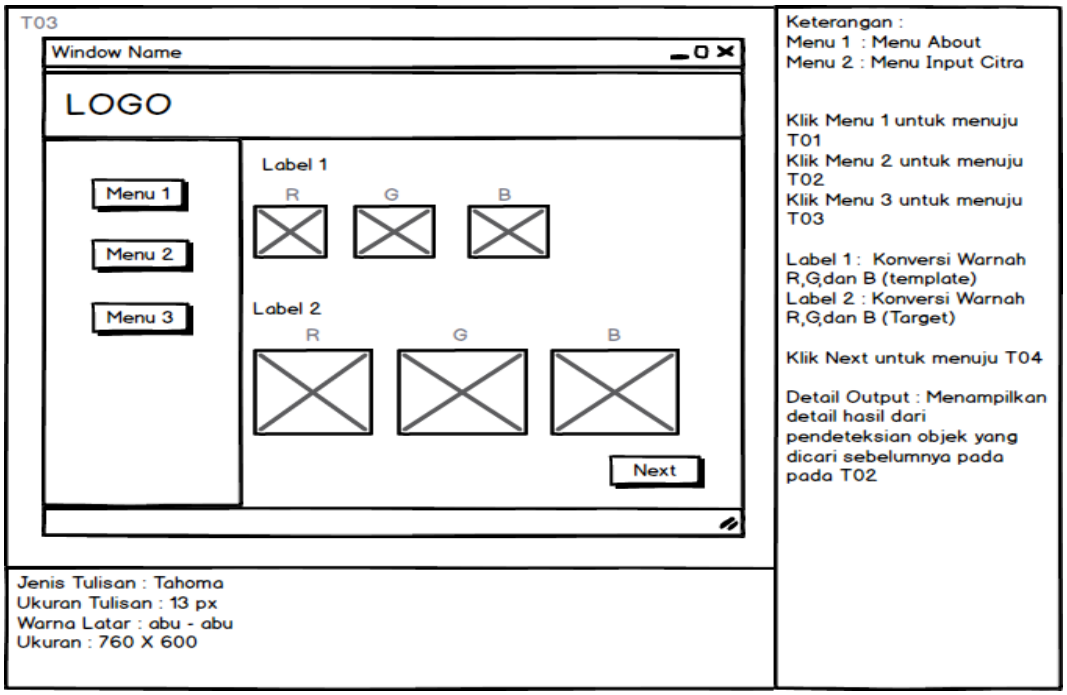
Perancangan antarmuka *input* citra merupakan tahapan rancangan antarmuka untuk menu pilihan ke dua. Perancangan antarmuka *input* citra dapat terlihat pada gambar 3.28.



Gambar 3. 28 Perancangan Antarmuka *Input* Citra

3.6.1.2.3 Perancangan Antarmuka Detail Output (T03)

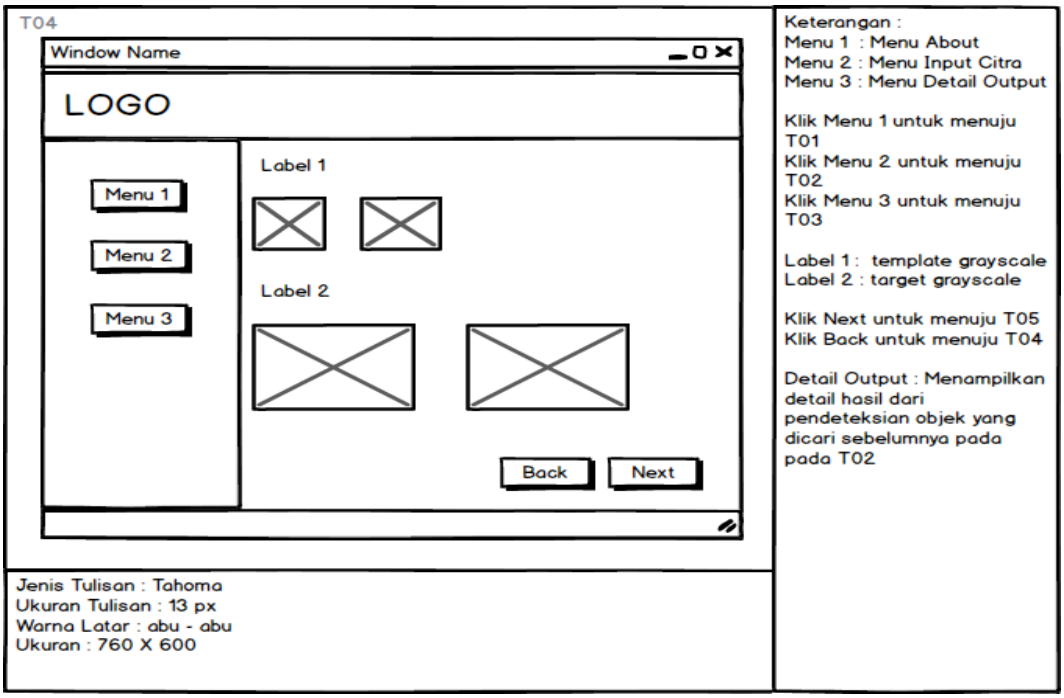
Perancangan antarmuka detail *output* merupakan tahapan rancangan antarmuka untuk menu pilihan ke tiga terdapat tiga perancangan didalamnya yaitu perancangan konversi warna r,g,b, konversi grayscale, dan perancangan hasil deteksi. Perancangan antarmuka hasil konversi r,g,b dapat terlihat pada gambar 3.29.



Gambar 3. 29 Perancangan Antarmuka konversi warna R,G,B

3.6.1.2.4 Perancangan Antarmuka Konversi Hasil Grayscale (T04)

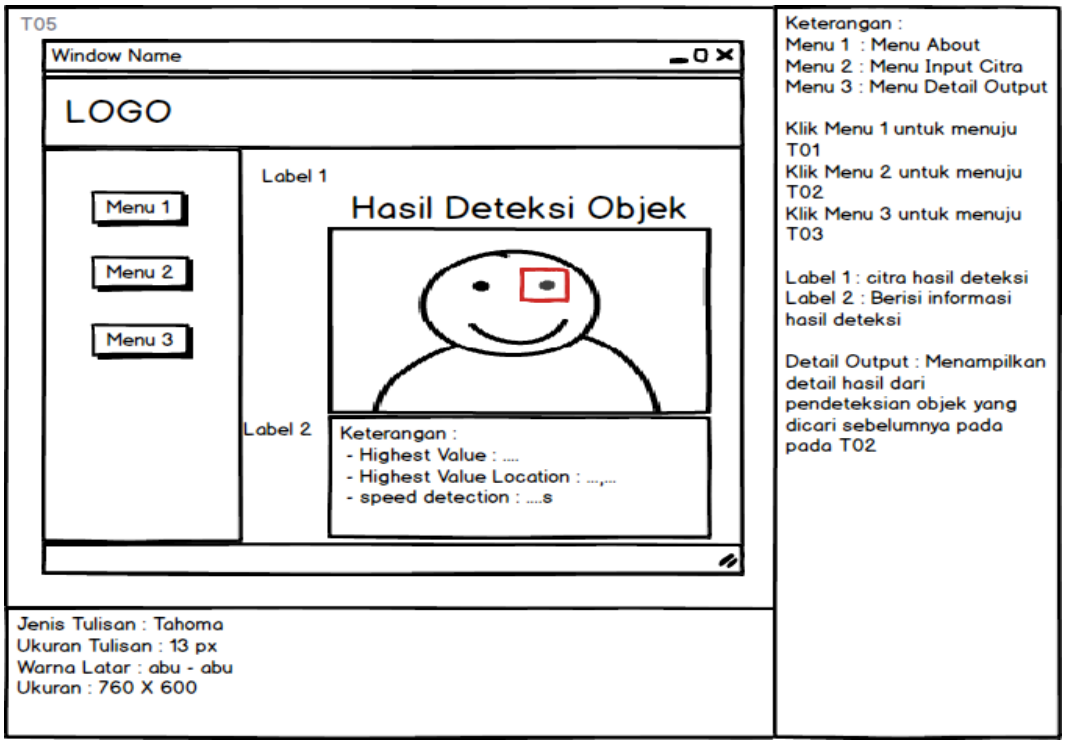
Perancangan antarmuka konversi *grayscale* merupakan rancangan antarmuka untuk melihat hasil dari konversi dari citra asli kedalam citra keabuan. perancangan antarmuka hasil *grayscale* dapat dilihat pada gambar 3.30.



Gambar 3. 30 Perancangan Antarmuka konversi grayscale

3.6.1.2.5 Perancangan Antarmuka Hasil Deteksi (T05)

Perancangan antarmuka *view result* merupakan rancangan antarmuka untuk menu pilihan ke tiga. Perancangan antarmuka *view result* dapat terlihat pada gambar 3.31.



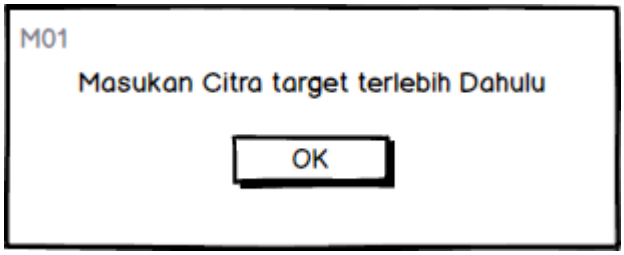
Gambar 3. 31 Perancangan Antarmuka Hasil Deteksi

3.6.1.3 Perancangan Pesan

Perancangan pesan merupakan tampilan dari suatu perangkat lunak yang berfungsi untuk menyampaikan notifikasi dan informasi kepada *user* agar perangkat lunak lebih interaktif. Perancangan pesan ini terdapat 4 perancangan pesan yaitu sebagai berikut :

3.6.1.3.1 Perancangan pesan jika target belum dipilih (M01)

Pesan ini muncul jika *user* menekan tombol *match process* dan sudah memilih *template* akan tetapi belum memilih citra target. Perancangan sistem ini dapat terlihat pada gambar 3.32.



Gambar 3. 32 Perancangan Pesan Jika Belum Memilih Citra Target

3.6.1.3.2 Perancangan Pesan jika template belum dipilih (M02)

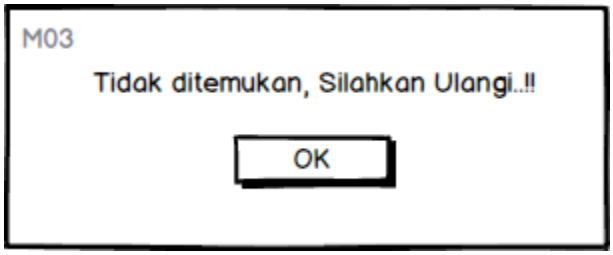
Pesan ini muncul jika *user* menekan tombol *match process* dan sudah memilih target akan tetapi belum memilih *template*. Perancangan sistem ini dapat terlihat pada gambar 3.33.



Gambar 3. 33 Perancangan Pesan Jika Belum Memilih Citra *Template*

3.6.1.3.3 Perancangan Pesan jika menu input citra belum dipilih (M04)

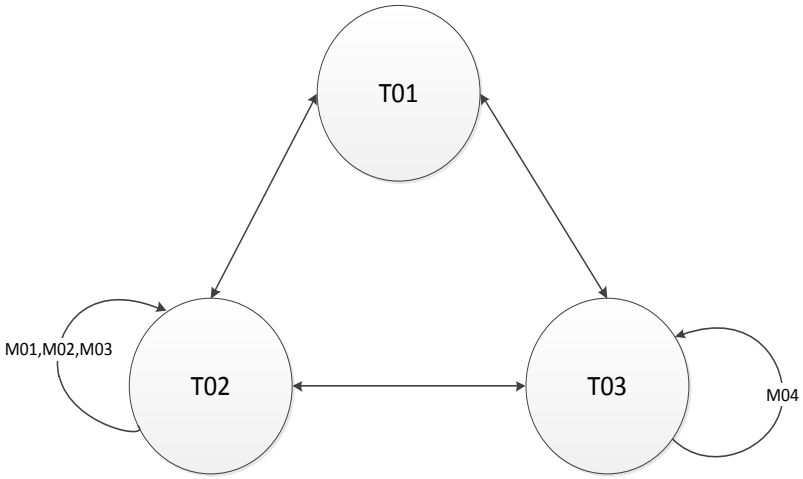
Pesan ini muncul jika *user* menekan menu *view result* akan tetapi belum memilih menu ke dua yaitu menu *input* citra terlebih dahulu. Perancangan sistem ini dapat terlihat pada gambar 3.34.



Gambar 3. 34 Perancangan Pesan Jika Belum Memilih Menu Input Citra

3.6.1.4 Perancangan Jaringan Semantik

Jairngan semantik adalah diagram yang menggambarkan alur-alur dari menu dan pesan dalam sebuah program. Jaringan semantik dari aplikasi pendeteksi objek pada citra digital yang dibangun yaitu seperti pada gambar 3.35.



Gambar 3. 35 Jaringan Semantik Aplikasi Pendeteksi Objek