BAB 3

ANALISIS DAN PERANCANGAN SISTEM

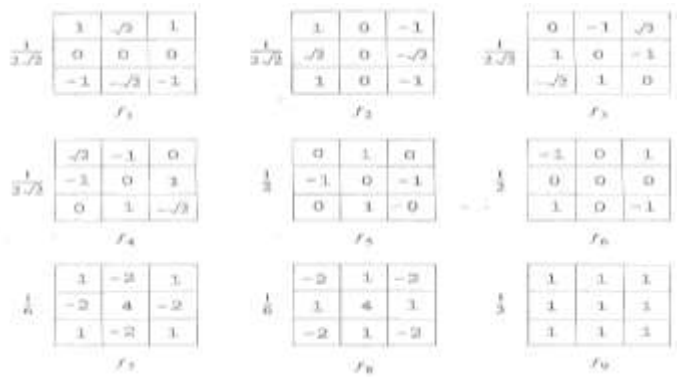
Analisis dan perancangan aplikasi memerlukan tahapan yang sistematis untuk mendapatkan aplikasi yang baik dan bersesuaian dengan tujuannya. Tahap awal dari analisis adalah menganalisis algoritma yang akan digunakan. Sedangkan untuk tahap perancangan aplikasi yaitu perancangan antarmuka.

3.1 Deteksi Tepi Menggunakan Algoritma Frei-Chen

Frei-Chen masks merupakan satuan basis *vector-vector*. Dapat diwakilkan pada setiap 3x3 *subimage* sebagai jumlah dari 9 *Frei-Chen masks.*

Setiap citra yang sudah ditentukan dilakukan proses projeksi yang mirip dengan proses *convolution*, dimana proses *convolution* adalah proses dimana gambar dimanipulasi dengan menggunakan masker eksternal untuk menghasilkan gambar yang baru.

Berikut ini adalah kesembilan gambar *Frei-Chen masks* tersebut:



Gambar 3.1 *Sembilan frei-chen mask*

Pada empat *mask* pertama digunakan untuk menentukan subruang garis tepi, empat *mask* berikutnya untuk menentukan subruang garis, dan *mask* yang terakhir untuk perhitungan rata-rata subruang. Untuk mendeteksi tepi, gambar diproyeksikan kedalam matrik yang ada. Persamaan proyeksi didefinisikan sebagai berikut:

Misalkan kita memiliki *subimage*, I_s:

$$I_s = \begin{bmatrix} a & b & a \\ a & b & a \\ a & b & a \end{bmatrix} \quad (3.1)$$

Untuk memproyeksikan *subimage* kedalam *frei-Chen masks*, mulai dengan melakukan perhitungan kedalam f_1 sampai ke f_9. Maka hasilnya sebagai berikut:

$$\frac{1}{2\sqrt{2}}\left[a(1) + b\left(\sqrt{2}\right) + a(1) + a(0) + b(0) + a(0) + a(-1) + b\left(-\sqrt{2}\right) + a(-1)\right] = c \quad (3.2)$$

Jika kita mengikuti proses dan proyeksi *subimage* I_s, kedalam setiap *Frei-Chen masks*, maka yang didapatkan adalah:

$f1 \rightarrow$ c, $f2 \rightarrow$ c, $f3 \rightarrow$ c, $f4 \rightarrow$ c, $f5 \rightarrow$ c, $f6 \rightarrow c$, $f7 \ldots$

$\rightarrow$ c ,$f8 \rightarrow$ c, $f9 \rightarrow$ c (3.3)

Berdasarkan hitungan, bobot yang tidak bernilai 0 adalah f5, f8 dan f9.

Dari hasil tersebut kemudian dikalikan dan dijumlahkan untuk mendapatkan deteksi tepi.

Secara garis besar, langkah-langkah algoritma pada proses deteksi tepi yang diterapkan pada citra:

1. Mulai memproyeksikan citra ke sub mask.
2. Lakukan konvolusi pada citra.
3. Kalikan produk konvolusi dengan mask.
4. Ambil berat (non-nol) dan kalikan lagi dengan topeng masing-masing sehingga menghasilkan jumlah yang sesuai.

Berikut algoritma metode *frei-chen*:

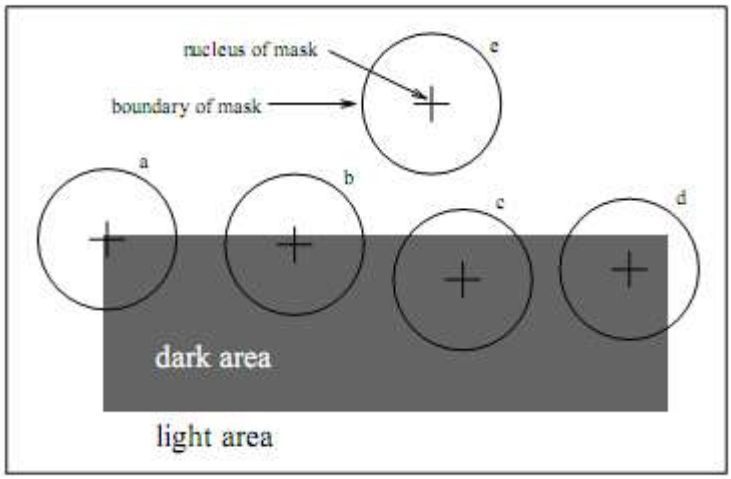
```
1    Procedure Deteksi_tepi_freichen (Input img : gambar,
       output img_deteksi_f : gambar)
2    {I.S : img untuk deteksi tepi}
3    {F.S : img_deteksi_f hasil deteksi tepi oleh frei-
       chen}
       Kamus
4    img = gmbr_yang_dipilih.ekstension;
       Algoritma
5      if size(varargin,2) > 0
6      then lower = varargin{1};
7    else if size(varargin,2) > 1
8    then upper = varargin{2};
       else
9    upper = 255;
10   end if
11   tresholded_img = zeros(img_size, 'uint8');
12   for x=1:img_size(1)
13   for y=1:img_size(2)
14   if img(x,y) >= lower && img(x,y) <=upper
15   then tresholded_img(x,y) = img(x,y);
16   end if
17   end for
18   end for
19   img = thresholded_img(x,y)
20   end if
21   img = cast(img, 'double');
22   filtered_img(:,:,1) = imfilter(img, freichen(:,:,1),
       'conv');
23   filtered_img(:,:,2) = imfilter(img, freichen(:,:,2),
       'conv');
24   filtered_img(:,:,3) = imfilter(img, freichen(:,:,3),
       'conv');
25   filtered_img(:,:,4) = imfilter(img, freichen(:,:,4),
       'conv');
26   img_deteksi_f    =    sqrt(filtered_img(:,:,1).^2    +
       filtered_img(:,:,2).^2 +     filtered_img(:,:,3).^2+
       filtered_img(:,:,4).^2);
```

Gambar 3.2 *Pseude Code frei-chen*

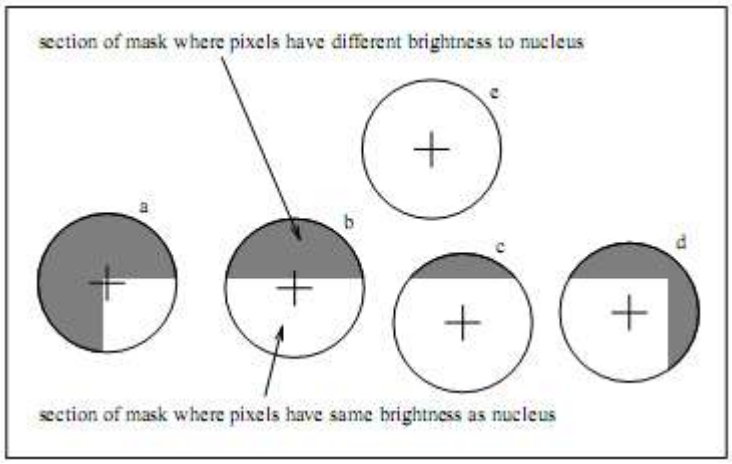
3.2 Deteksi Tepi Menggunakan Algoritma SUSAN (*Smallest Univalue Segment Assimilation Nucleus*)

Teori *SUSAN* menggunakan metoda yang biasa digunakan dalam proses pendeteksian sisi yaitu dengan menggunakn matrik *mask* yang telah ditentukan aturan-aturannya (*rules*). Jika *brightness* dari setiap pixel di dalam *mask* tersebut dibandingkan dengan *brightness* dari inti *mask* (pixel yang berada ditengah matrik) tersebut, maka sebagian atau seluruh area dari *mask* dapat didefinisikan memiliki *brightness* yang sama atau mirip dengan nilai *brightness* yang dimiliki oleh inti *mask* atau *nucleus*. Konsep bahwa setiap titik dari citra dihubungkan dengan sebuah area lokal yang memiliki *brightness* yang hampir sama adalah dasar dari teori *SUSAN* [3].

Area ini kemudian dikenal sebagai *USAN* (*Univalue Segment Assimilating Nucleus)* dan berisi informasi mengenai struktur dari sebuah citra. Berdasarkan area ini maka bentuk dua dimensi atau sudut dan sisi dari sebuah citra dapat di deteksi. Seperti yang terlihat pada gambar dibawah ini, area *USAN* digambarkan pada bagian putih dari *mask.*



Gambar 3.3 *Lima circular masks dengan posisi yang berbeda pada sebuah citra*



Gambar 3.4 *Empat circular masks dengan warna yang serupa, USAN ditampilkan sebagai bagian putih dari mask*

Area dari sebuah *USAN* menggambarkan informasi penting mengenai struktur dari sebuah citra pada setiap area yang telah dipetakan. *USAN* area bernilai maksimum ketika *nucleus* berada pada bagian datar dari permukaan citra. Nilainya turun sampai setengahnya ketika mendekati sisi dan turun lebih besar

lagi ketika berada didalam sudut citra. Nilai dari area *USAN* inilah yang digunakan sebagai penentu posisi *edge*.

Pendeteksian sisi *SUSAN* diimplementasikan dengan menggunakan circular *mask* yang terdiri dari 37 pixel atau *mask 3x*. Matrik *mask* diletakkan pada setiap bagian dari citra. Untuk setiap bagian tersebut *brightness* dari setiap pixel didalam matrik dibandingkan dengan pixel *nucleus.* Jika perbedaan nilai *brightness* dari tiap pixel kurang dari nilai threshold ”t”, maka keluaran dari pembandingan tersebut adalah 1 dan jika sebaliknya maka nilai 0. Aturan tersebut dituliskan ke dalam persamaan sebagai berikut :

$$c(\vec{r}, \overrightarrow{r0}) = \begin{cases} 1 , \left| I(\vec{r}) - I\left(\overrightarrow{r0}\right)\right| \leq t \\ 0 , \left| I(\vec{r}) - I\left(\overrightarrow{r0}\right)\right| > t \end{cases} \quad (3.4)$$

Keterangan:

t = nilai *threshold*

$\vec{r_0}$ = posisi *nucleus* pada citra

$\vec{r}$ = posisi *nuclues* pada poin citra yang lain

Jika proses pembandingan telah dilakukan pada setiap pixel, maka akan diperoleh besar area *SUSAN* dengan menjumlahkan pixel-pixel didalam *window*

mask yang bernilai 1. Nilai area *SUSAN* diperoleh berdasarkan persamaan berikut :

$$n(\overrightarrow{r0}) = \sum_{\vec{r}} c(\vec{r},\overrightarrow{r0}) \quad (3.5)$$

Keterangan:

n = total *ouput*

c = *output* perbandingan

Untuk mendapatkan respon *edge*, proses selanjutnya adalah membandingkan nilai *n* dengan tetapan threshold *g* (*geometric threshold)* yang bernilai 3*Nmax/4, dimana Nmax adalah nilai maksimum yang dapat dicapai oleh *n.* Nilai respon awal dari *edge* didapatkan dengan menggunakan aturan berikut :

$$R(\overrightarrow{r0}) = \begin{cases} g - n(\overrightarrow{r0}), n(\overrightarrow{r0}) < g \\ 0 \quad \textit{otherwise}, \end{cases} \quad (3.6)$$

Keterangan:

$R(\overrightarrow{r0})$ = hasil garis tepi

g = *geometric threshold*

n = total *output*

Formulasi ini merupakan metoda *SUSAN*, dimana semakin kecil area *SUSAN* maka akan semakin besar respon sisi yang dihasilkan.

Secara garis besar, langkah-langkah algoritma pada proses deteksi tepi yang diterapkan pada citra:

1. Letakkan sebuah *mask* circular pada pixel yang akan diuji.
2. Hitung jumlah pixel didalam circular *mask* yang memilki *brigtness* yang mirip dengan *brightness nucleus.*
3. Kurangi ukuran SUSAN dari *threshold geometric* untuk menghasilkan citra

```
1   Procedure Deteksi_tepi_freichen (Input IMG : gambar,
    output img_deteksi_s : gambar)
2   {I.S  : IMG untuk deteksti tepi}
3         {F.S  : img_deteksi_s hasil deteksi tepi oleh
    SUSAN}
    Kamus
4   IMG = gmbr_yang_dipilih.ekstension;
    Algoritma
5   IMG = IMG + 255;
6   [m,n] = size(IMG);
7    KW = (2 * KR) + 1;
8    KM = ones(KW,KW);
9    for i = -KR:KR
10   for j = -KR:KR
11   if (round(sqrt((i.^2) + (j.^2))) > KR)
12   KM(i+KR+1,j+KR+1) = 0;
13   End if
14   End for
15   End for
16   KAREA = nnz(KM);
17   GT = 3 * KAREA / 4;
18   IMG                                                =
    [repmat(IMG(1,:),KR,1);IMG;repmat(IMG(m,:),KR,1)];
19   IMG                                                =
    [repmat(IMG(:,1),1,KR),IMG,repmat(IMG(:,n),1,KR)];
20   EDG = zeros(m,n);
21   for i = KR+1:m+KR
22   for j = KR+1:n+KR
23   USAN  =  KM  .*  exp(-((IMG(i-KR:i+KR,  j-KR:j+KR)  -
     IMG(i,j)) / TR).^6);
24   if (OP)
25   USAN_BINARY = ceil(USAN);
26  if (nnz(USAN_BINARY) < (KAREA - KR))
27   USAN = bwselect(USAN_BINARY,KR+1,KR+1,8) .* USAN;
28   End if
```

```
34   End for
35   End for
```

Gambar 3.5 *Pseude Code SUSAN*

3.3 Gray – Scale

Gray-scale atau skala keabu-abuan, adalah proses umum untuk meratakan ketiga kanal citra yaitu kanal *RGB (Red, Green, Blue).* Tujuan agar proses perhitungan yang melibatkan operasi perhitungan citra yang lainnya dapat lebih efisien. Berikut ini adalah persamaannya:

$$bit_R(x,y) = 0{,}11 \times bit_R(x,y) + 0{,}59 \times bit_G(x,y) + 0{,}3 \times bit_B(x,y) \quad (3.7)$$

dimana,

$$bit_G(x,y) = bit_B(x,y) = bit_R(x,y) \quad (3.8)$$

3.4 Analisis Kebutuhan Data Non Fungsional

Analisis kebutuhan ini merupakan pendukung aplikasi yang akan dijalankan. Adapun kebutuhan-kebutuhannya yaitu:

3.4.1 Kebutuhan Perangkat Keras

Berikut ini adalah spesifikasi perangkat keras yang digunakan dalam penelitian ini yaitu:

a. Processor : 1,6 GHz.
b. Memory : 2048 MB.
c. VGA : Intel X3150 256MB.
d. HDD : 320 GB.
e. Monitor : 10.1 inch.
f. Mouse dan Keyboard.

3.4.2 Kebutuhan Perangkat Lunak

Perangkat lunak digunakan dalam sebuah sistem untuk memberi perintah kepada perangkat keras agar bisa saling berinteraksi diantara keduanya. Perangkat lunak yang dibutuhkan dalam membangun aplikasi ini adalah sebagai berikut :

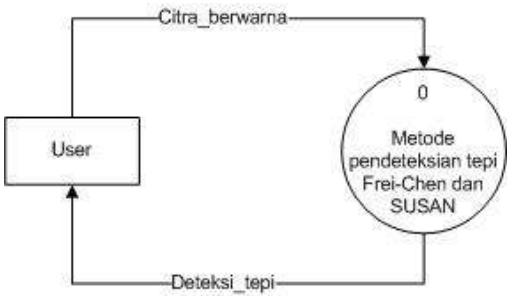
a. Sistem Operasi Windows.
b. Matlab.

3.5 Kebutuhan Fungsional

Proses-proses yang terjadi di dalam aplikasi ini akan dijelaskan sebagai berikut:

3.5.1 Diagram Konteks

Diagram ini adalah diagram level tertinggi dari DFD yang menggambarkan hubungan sistem dengan lingkungan luar. Diagram konteks dari sistem yang dirancang adalah sebagai berikut:



Gambar 3.6 *Diagram Konteks Analisis Deteksi Tepi*

Proses yang terjadi pada Diagram konteks tersebut dapat dijelaskan melalui spesifikasi proses sebagai berikut:

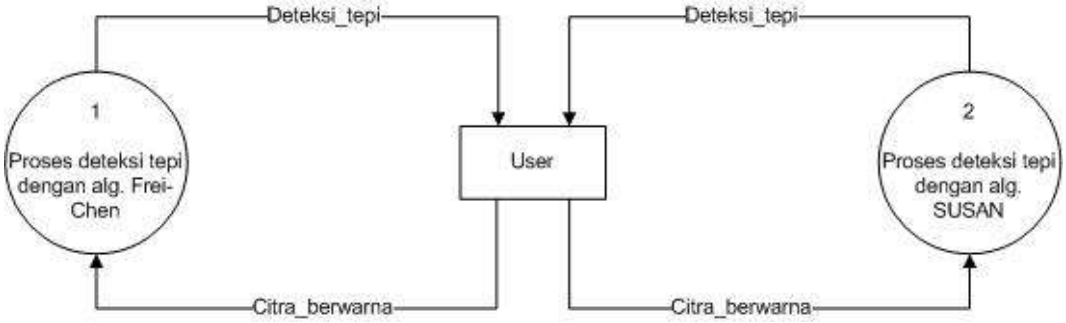
Tabel 3.1 Spesifikasi Proses Diagram Konteks

No	Nama Proses	Input	Keterangan Proses	Output
0	Metode pendeteksian tepi Frei-Chen dan SUSAN	Citra_berwarna	Mendeteksi tepi sebuah citra dengan metode Frei-Chen dan SUSAN	Deteksi_tepi

Selanjutnya Diagram Konteks diatas dapat dijabarkan lebih luas lagi dengan memecah proses Diagram Konteks kedalam proses-proses yang lebih kecil.

3.5.2 Data Flow Diagram

Data Flow Diagram (DFD) merupakan pemodelan terstruktur yang berfungsi untuk menggambar sistem sebagai suatu jaringan fungsional yang dihubungkan satu sama lain dengan alur data baik secara manual maupun komputerisasi. Penjabaran pertama disebut dengan DFD level 0, berikut gambar DFD level 0.



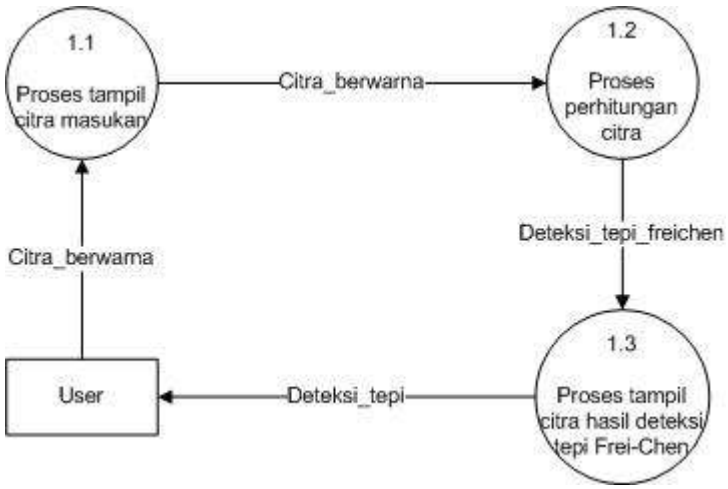
Gambar 3.7 *DFD Level 0*

Penjelasan mengenai proses-proses yang terjadi pada DFD Level 0 tersebut diperlihatkan dalam tabel berikut:

Tabel 3.2 Spesifikasi Proses DFD Level 0

No	Nama Proses	Input	Keterangan	Output
1	Proses deteksi tepi dengan alg. Frei-Chen	Citra_berwarna	Mendeteksi tepi dengan alg. Frei-Chen	Deteksi_tepi_freichen
2	Proses deteksi tepi dengan alg. SUSAN	Citra_berwarna	Mendeteksi tepi dengan alg. SUSAN	Deteksi_tepi

Proses 1 dari DFD level 0 diatas terdiri atas 3 (tiga) proses utama yang terjadi didalamnya, aliran data yang terjadi pada proses ini sebagai DFD Level 1 proses Deteksi Tepi Frei-Chen, diperlihatkan pada gambar berikut:



Gambar 3.8 *DFD Level 1 Proses 1 Deteksi Tepi Frei-Chen*

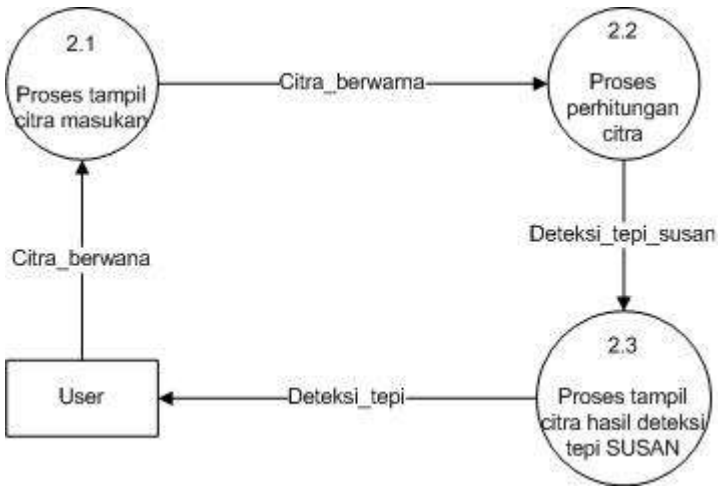
Untuk spesifikasi proses yang terjadi pada DFD Level 1 proses 1 ini ditujukan pada tabel.

Tabel 3.3 Spesifikasi Proses DFD Level 1 Proses 1 Deteksi Tepi Frei-Chen

No	Nama Proses	Input	Keterangan Proses	Output
1.1	Proses tampil citra masukan	Citra_berwarna	Mencari sebuah data citra(masukan) dan menampilkannya	Citra_berwarna
1.2	Proses perhitungan citra	Citra_berwarna	Melakukan perhitungan dengan	Deteksi_tepi_freichen

			alg. Frei-Chen agar mendapat deteksi tepi pada citra	
1.3	Proses tampil citra hasil deteksi tepi Frei-Chen	Deteksi_tepi_freichen	Menampilkan citra yang telah diproses dengan metode pendeteksian tepi Frei-Chen	Deteksi_tepi

Untuk penjelasan proses 2 deteksi tepi SUSAN, juga terdapat 3 (tiga) proses utama. Berikut gambar penjabaran DFD Level 1 proses 2.



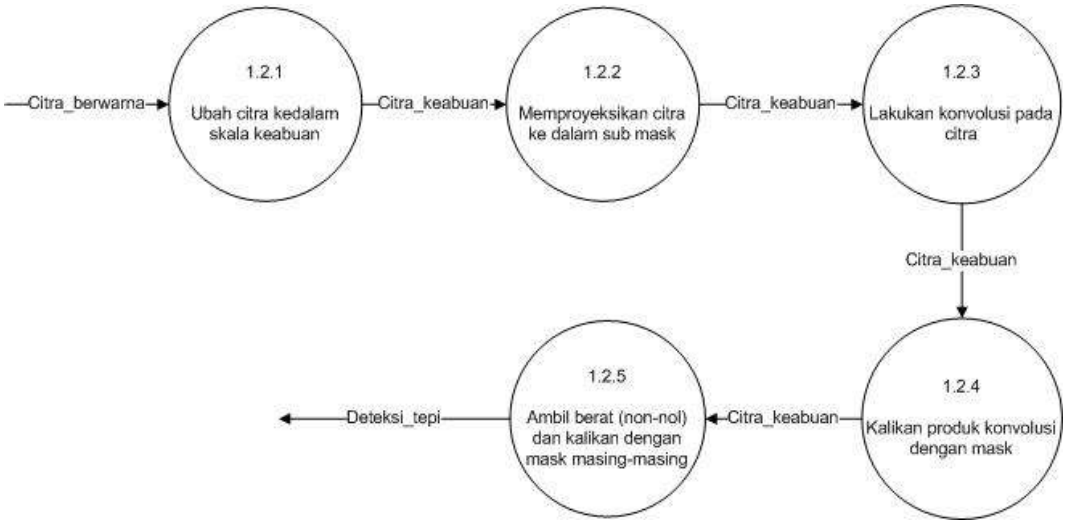
Gambar 3.9 *DFD Level 1 proses 2 Deteksi Tepi SUSAN*

Berikut tabel spesifikasi proses dari DFD Level 1 proses 2:

Tabel 3.4 Spesifikasi Proses DFD Level 1 proses 2 Deteksi Tepi SUSAN

No	Nama Proses	Input	Keterangan Proses	Output
2.1	Proses tampil citra masukan	Citra_berwarna	Mencari sebuah data citra(masukan) dan menampilkannya	Citra_berwarna
2.2	Proses perhitungan citra	Citra_berwarna	Melakukan perhitungan dengan alg. SUSAN agar mendapat deteksi tepi pada citra	Deteksi_tepi_susan
2.3	Proses tampil citra hasil deteksi tepi SUSAN	Deteksi_tepi_susan	Menampilkan citra yang telah diproses dengan metode pendeteksian tepi SUSAN	Deteksi_tepi

Proses 1.2 dari DFD level 1 diatas terdiri atas 5 (lima) proses utama yang terjadi didalamnya, aliran data yang terjadi pada proses ini sebagai DFD Level 2 proses Deteksi Tepi Frei-Chen, diperlihatkan pada gambar berikut:



Gambar 3.10 *DFD level 2 proses 1.2 Deteksi tepi Frei-Chen*

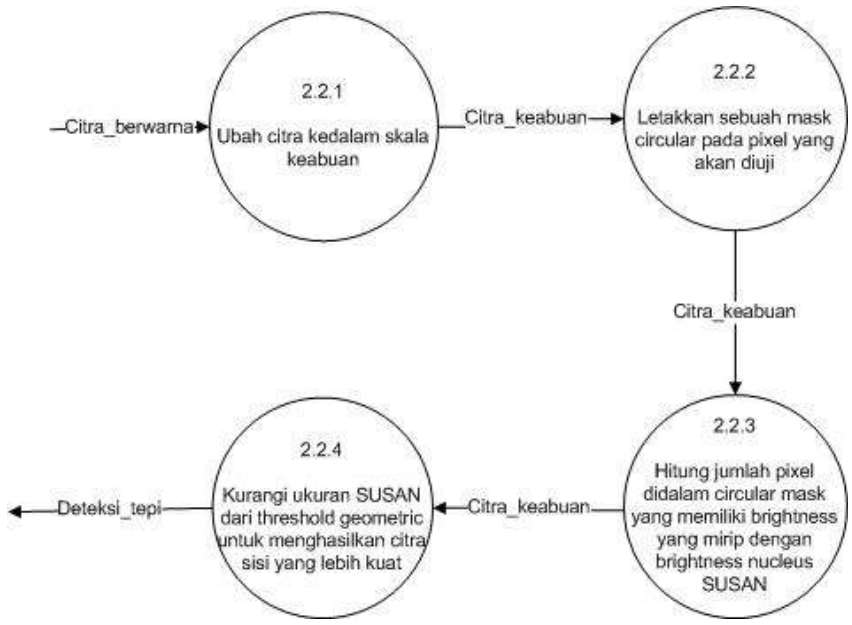
Berikut tabel spesifikasi proses dari DFD Level 2 proses 1.2:

Tabel 3.5 Spesifikasi Proses DFD Level 2 proses 2 Deteksi Tepi Frei-Chen

No	Nama Proses	Input	Keterangan Proses	Output
1.2.1	Ubah Citra kedalam skal keabuan	Citra_berwarna	Melakukan perubahan citra berwarna kedalam skala keabuan	Citra_keabuan
1.2.2	Memproyeksikan citra kedalam sub mask	Citra_keabuan	Melakukan proyeksi citra kedalam sub mask	Citra_keabuan
1.2.3	Melakukan konvolusi pada citra	Citra_keabuan	Melakukan konvolusi pada citra digital dengan filter frei chen	Citra_keabuan

1.2.4	Kalikan produk konvolusi dengan mask	Citra_keabuan	Melakukan perhitungan perkalian	Citra_keabuan
1.2.5	Ambil berat (non nol) dan kalikan dengan mask masing-masing	Citra_keabuan	Melakukan perhitungan perkalian dengan mask masing-masing setelah pengambilan nilai non nol.	Deteksi_tepi

Proses 2.2 dari DFD level 1 diatas terdiri atas 4 (lima) proses utama yang terjadi didalamnya, aliran data yang terjadi pada proses ini sebagai DFD Level 2 proses Deteksi Tepi SUSAN, diperlihatkan pada gambar berikut:



Gambar 3.11 DFD *level 2 proses 2.2 Deteksi tepi SUSAN*

Berikut tabel spesifikasi proses dari DFD Level 2 proses 2.2:

Tabel 3.6 Spesifikasi Proses DFD Level 2 proses 2 Deteksi Tepi SUSAN

No	Nama Proses	Input	Keterangan Proses	Output
2.2.1	Ubah Citra kedalam skal keabuan	Citra_berwarna	Melakukan perubahan citra berwarna kedalam skala keabuan	Citra_keabuan
2.2.2	Letakkan sebuah mask circular pada pixel yang akan diuji	Citra_keabuan	Melakukan proyeksi citra kedalam mask circular	Citra_keabuan
2.2.3	Hitung jumlah piksel didalam circular mask yang memiliki brightness yang mirip dengan brightness nucleus SUSAN	Citra_keabuan	Melakukan perhitungan jumlah piksel didalam circular mask yang memilik tingkat kecerahan mirip dengan tingkat kecerahan SUSAN	Citra_keabuan
2.2.4	Kurangi ukuran SUSAN dari threshold geometric untuk menghasilkan citra	Citra_keabuan	Melakukan perhitungan pengurangan untuk menghasilkan citra sisi yang lebih kuat	Deteksi_tepi

	sisi yang lebih kuat			

3.5.3 Perancangan Kamus Data

Kamus data merupakan data yang disusun untuk memudahkan proses analisis dan desain. Dengan kamus data pemakai dan analisis sistem akan memiliki pemahaman umum mengenai input, output, dan komponen penyimpanan. Dibawah ini merupakan tabel yang memperlihatkan kamus data untuk menjelaskan data pada DFD yang telah dijelaskan sebelumnya.

Tabel 3.7 Perancangan Kamus Data

No	Nama	Deskripsi
1	Citra_warna	Citra_warna= file citra asli
2	Citra_keabuan	Hasil perubahan intensitas warna pada citra
3	Data_tepi	Hasil deteksi tepi citra dengan algoritma Frei-Chen dan SUSAN

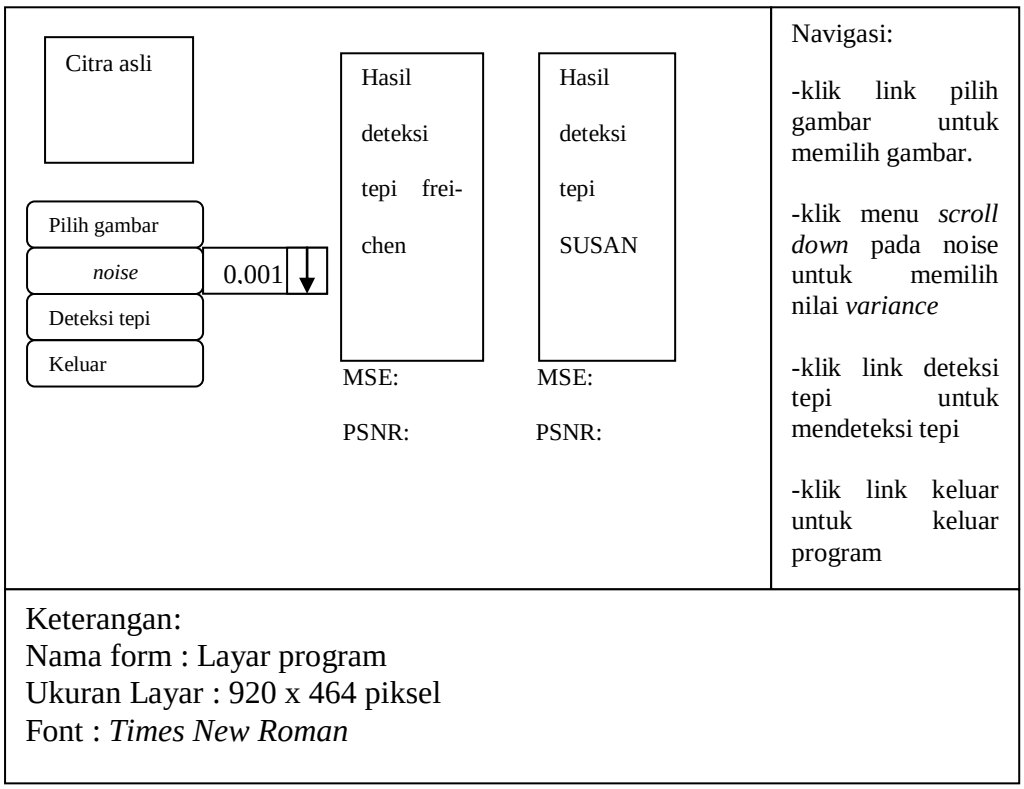
3.5.4 Perancangan Antarmuka Pemakai (*User Interface*)

Antarmuka pemakai (*User Interface*) adalah aspek program yang dapat dilihat oleh pengguna, dan perintah-perintah atau mekanisme yang digunakan pemakai untuk mengendalikan operasi dan memasukkan data.

Berikut ini adalah perancangan antarmuka aplikasi Pendeteksian Tepi Citra yang dirancang dalam pemrograman Matlab.

3.5.4.1 Antarmuka Pendeteksian Tepi

Tampilan utama merupakan tampilan awalan yang akan menghubungkan *user* dengan fitur utama perangkat lunak. Oleh karena itu, untuk tampilan utama ini dirancang dengan tampilan sederhana untuk mempermudah *user* dalam berinteraksi dengan sistem.



Gambar 3.12 *Tampilan program*

3.5.5 Perancangan Prosedural

Pada bagian ini akan dibuat deskripsi rinci dari perangkat lunak, yaitu algoritma yang akan digunakan untuk mendeteksi suatu citra digital. Deskripsi rinci tersebut adalah sebagai berikut:

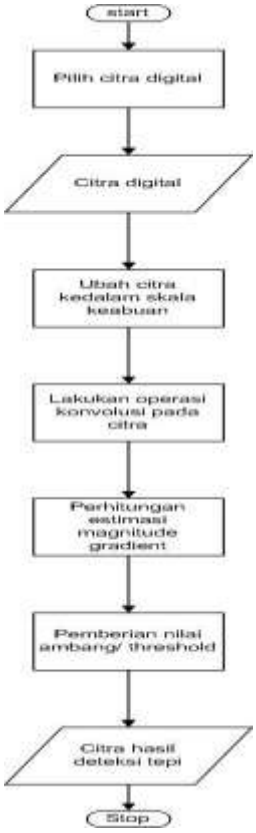
3.5.5.1 Prosedur Algoritma Frei-Chen

Berikut ini prosedur dalam melakukan deteksi tepi pada citra digital dengan menggunakan algoritma frei-chen:

1. Pilih gambar yang akan dideteksi tepinya.
2. Lakukan perubahan warna pada gambar dengan skala keabuan dengan persamaan (3.8).
3. Lakukan filter dengan operasi konvolusi pada citra dengan *mask* frei-chen (gambar 3.1).
4. Lalu estimasi magnitude gradient untuk tiap piksel menggunakan persamaan:

$$M = \sqrt{Ix^2 + Iy^2} \tag{3.9}$$

5. Dengan memberikan nilai ambang atau *threshold* terhadap magnitude gradient, maka didapat citra yang berisi hasil dari deteksi tepi.



Gambar 3.13 *Flowchart frei-chen*

3.5.5.2 Prosedur Algoritma SUSAN

Berikut ini prosedur dalam melakukan deteksi tepi pada citra digital dengan menggunakan algoritma SUSAN:

1. Pilih gambar yang akan dideteksi tepinya.
2. Lakukan perubahan warna pada gambar dengan skala keabuan dengan persamaan (3.7).

3. Letakkan *mask* circular pada piksel yang akan diuji, berikut persamaan *mask* yang digunakan:

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} \quad (3.10)$$

4. Lakukan perbandingan antara *brightness* pada citra dengan *mask* circular. Dengan menerapkan persamaan (3.4).

5. Jumlahkan piksel –piksel dari hasil perbandingan yang bernilai 1, maka dapat diporeleh nilai area SUSAN berdasarkan persamaan (3.5).

6. Lakukan perbandingan nilai **n** dengan tetapan *threshold* **g** yang bernilai 3*Nmax/4, dengan menggunakan persamaan (3.6).



Gambar 3.14 *Flowchart SUSAN*