BAB 4

IMPLEMENTASI DAN PENGUJIAN

Tahap implementasi merupakan tahap lanjutan dari tahap perancangan sistem. Tahap ini juga merupakan tahap untuk meletakan sistem agar siap untuk dioperasikan dan dipandang sebagai usaha untuk mewujudkan sistem yang telah dirancang.

4.1 Perangkat Lunak Pembangun

Perangkat lunak yang digunakan pada sistem komputer yang digunakan untuk membangun aplikasi deteksi tepi adalah sebagai berikut:

1. Sistem Operasi *Windows 7*.
2. Matlab.

4.2 Perangkat Keras Pembangun

Kebutuhan perangkat keras (*hardware)* yang diperlukan untuk mengimplementasikan program aplikasi yang dibuat. Berikut kebutuhan perangkat keras (*hardware)* yang digunakan yaitu:

1. Processor: Intel Atom 1.6 GHZ.
2. Memory: 2048 MB.
3. Hardisk: 320 GB.
4. VGA:Intel GMA 3100.
5. Monitor: 10 Inchi.

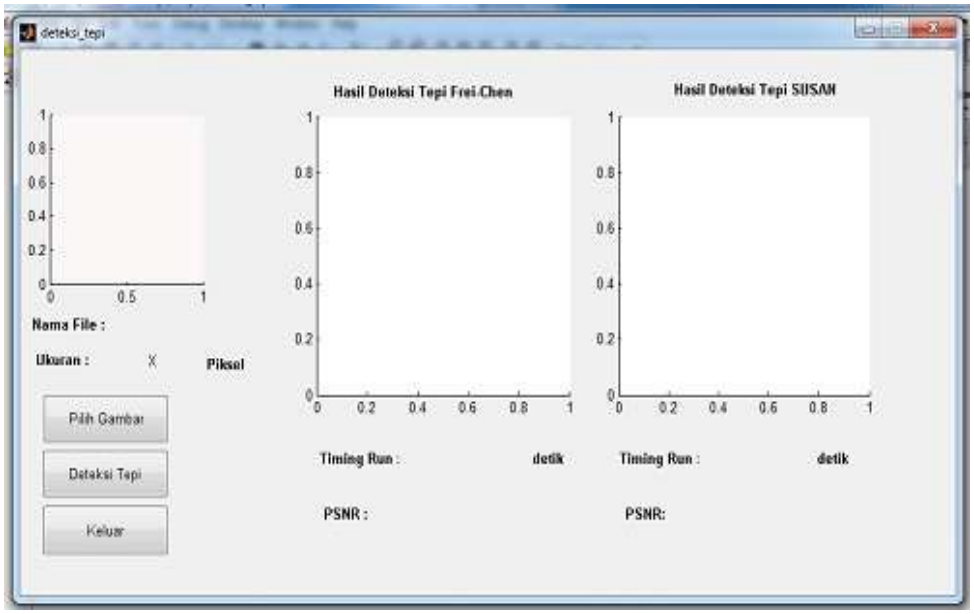
4.3 Implementasi Program

4.3.1 Implementasi Antarmuka

Antarmuka merupakan saran dialog atau interaksi antara pemakai (user) dengan sistem.

4.3.1.1 Form Utama

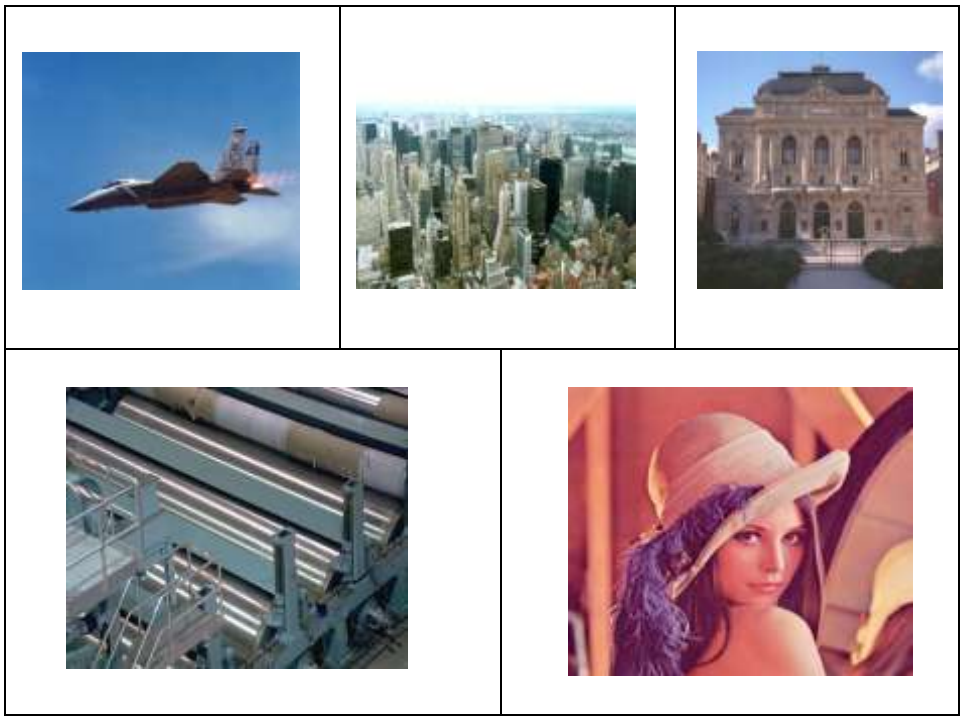
Pada form utama ini terdapat program secara keseluruhan beserta dengan prosedur-prosedur programnya. Tampilan dari form utama ini dapat dilihat pada gambar 4.1 dibawah ini:



Gambar 4.1 *Antarmuka Aplikasi*

4.4 Uji Coba Program

Pada sub bab ini akan dibahas tentang uji coba dari program yang telah dibuat menyangkut masukan (*input),* proses dan hasil dari program *(output).* Ada 5 citra yang dijadikan bahan uji, berikut citra yang dijadikan bahan uji:



Gambar 4.2 *Lima macam citra digital*

4.5 Pengujian *White-Box*

4.5.1 *Procedure Sort Frei-Chen dan SUSAN*

Pada gambar 4.3 dan 4.4 merupakan *procedure sort* yang digunakan pada masing-masing algoritma

```
1   img=gmbr_yang_dipilih.ekstension;
2   if size(varargin,2) > 0
3         then lower = varargin{1};
4      else if size(varargin,2) > 1
5       then upper = varargin{2};
        else
6         upper = 255;
7       end if
8   tresholded_img = zeros(img_size, 'uint8');
9    for x=1:img_size(1)
10          for y=1:img_size(2)
11            if img(x,y) >= lower && img(x,y) <=upper
12               then tresholded_img(x,y) = img(x,y);
13                end if
14             end for
```

```
15          end for
16      img = thresholded_img(x,y)
17           end if
18      img = cast(img, 'double');
19      filtered_img(:,:,1)          =          imfilter(img,
   freichen(:,:,1), 'conv');
20           filtered_img(:,:,2)        =       imfilter(img,
   freichen(:,:,2), 'conv');
21           filtered_img(:,:,3)        =       imfilter(img,
   freichen(:,:,3), 'conv');
22           filtered_img(:,:,4)        =       imfilter(img,
   freichen(:,:,4), 'conv');
23      edges      =      sqrt(filtered_img(:,:,1).^2      +
        filtered_img(:,:,2).^2  +    filtered_img(:,:,3).^2+
        filtered_img(:,:,4).^2);
```

Gambar 4.3 *Procedure sort frei-chen*

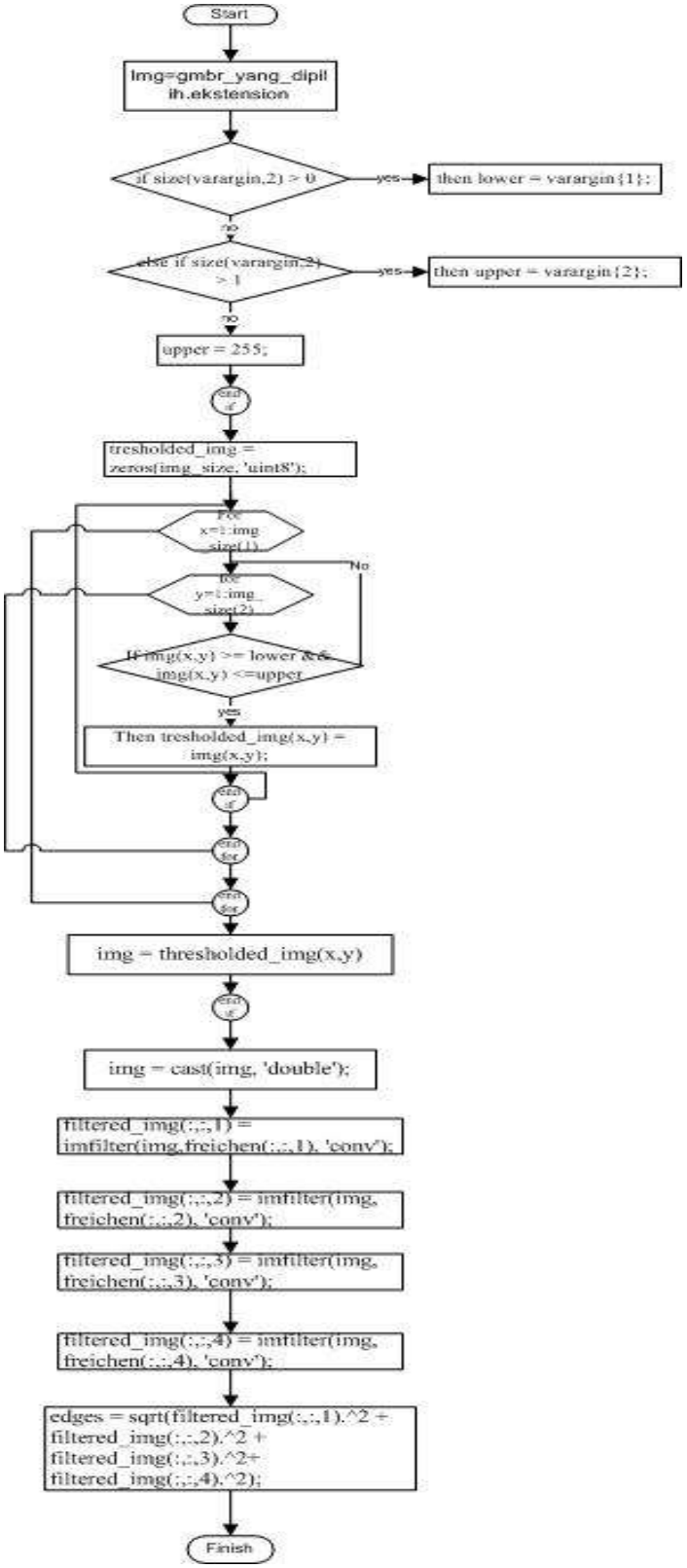
```
1   IMG=gmbr_yang_dipilih.ekstension;
2   IMG = IMG + 255;
3   [m,n] = size(IMG);
4    KW = (2 * KR) + 1;
5    KM = ones(KW,KW);
6       for i = -KR:KR
7           for j = -KR:KR
8               if (round(sqrt((i.^2) + (j.^2))) > KR)
9                   KM(i+KR+1,j+KR+1) = 0;
10              End if
11          End for
12      End for
13     KAREA = nnz(KM);
14      GT = 3 * KAREA / 4;
15      IMG                                                    =
    [repmat(IMG(1,:),KR,1);IMG;repmat(IMG(m,:),KR,1)];
16      IMG                                                    =
    [repmat(IMG(:,1),1,KR),IMG,repmat(IMG(:,n),1,KR)];
17      EDG = zeros(m,n);
18      for i = KR+1:m+KR
19          for j = KR+1:n+KR
20    USAN = KM .* exp(-((IMG(i-KR:i+KR, j-KR:j+KR) - IMG(i,j))
              / TR).^6);
21  if (OP)
22   then  USAN_BINARY = ceil(USAN);
```

```
23  Else if (nnz(USAN_BINARY) < (KAREA - KR))
24      USAN = bwselect(USAN_BINARY,KR+1,KR+1,8) .* USAN;
25    End if
26   End if
27   USAN_AREA = sum(sum(USAN));
28   if (USAN_AREA < GT)
29       EDG(i-KR,j-KR) = GT - USAN_AREA;
30       End if
31     End for
32   End for
```

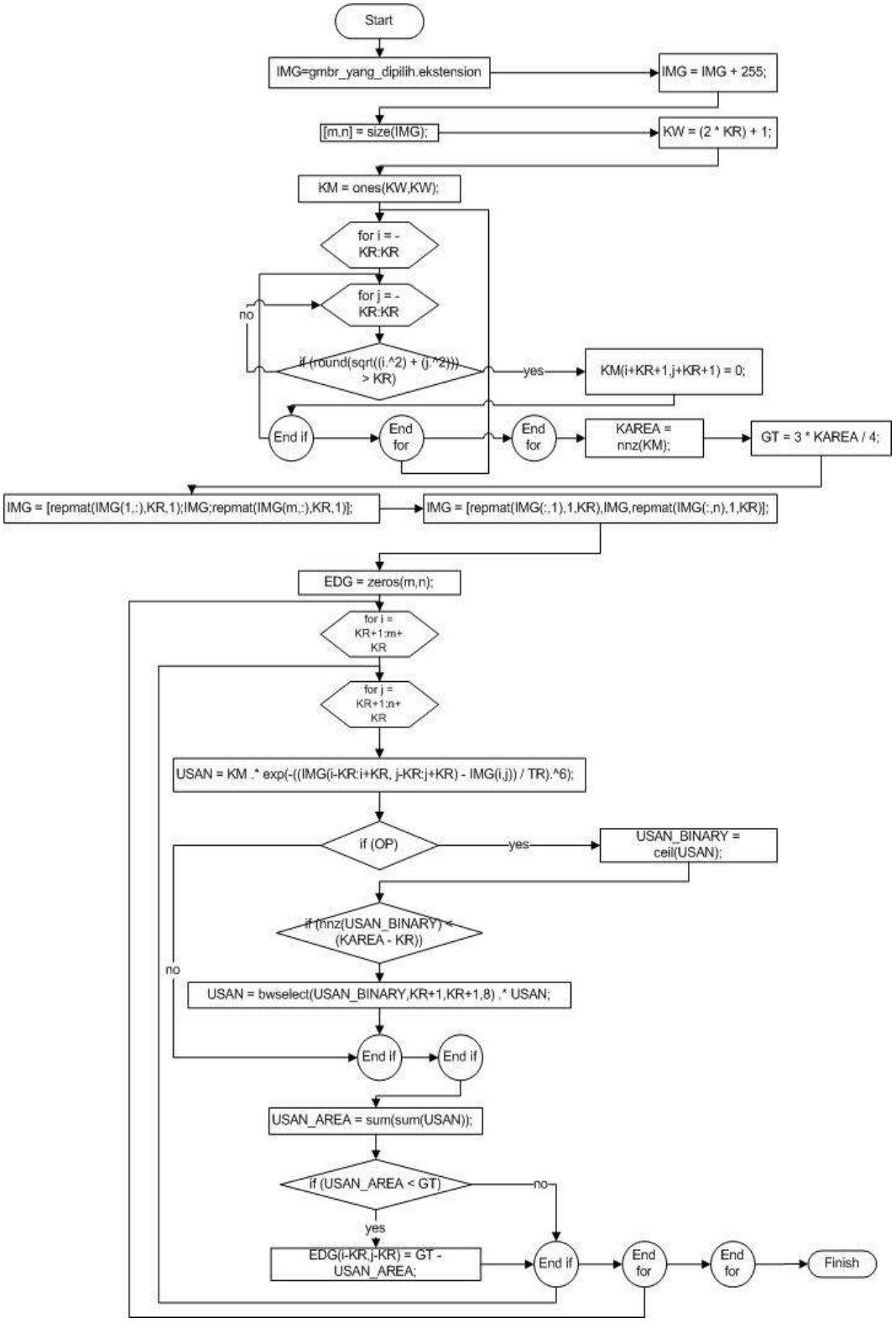
Gambar 4.4 *Procedure sort susan*

4.5.2 *Flowchart Frei-Chen dan SUSAN*

Pada Gambar 4.5 dan 4.6 merupakan *flowchart* yang dimiliki pada masing-masing algoritma



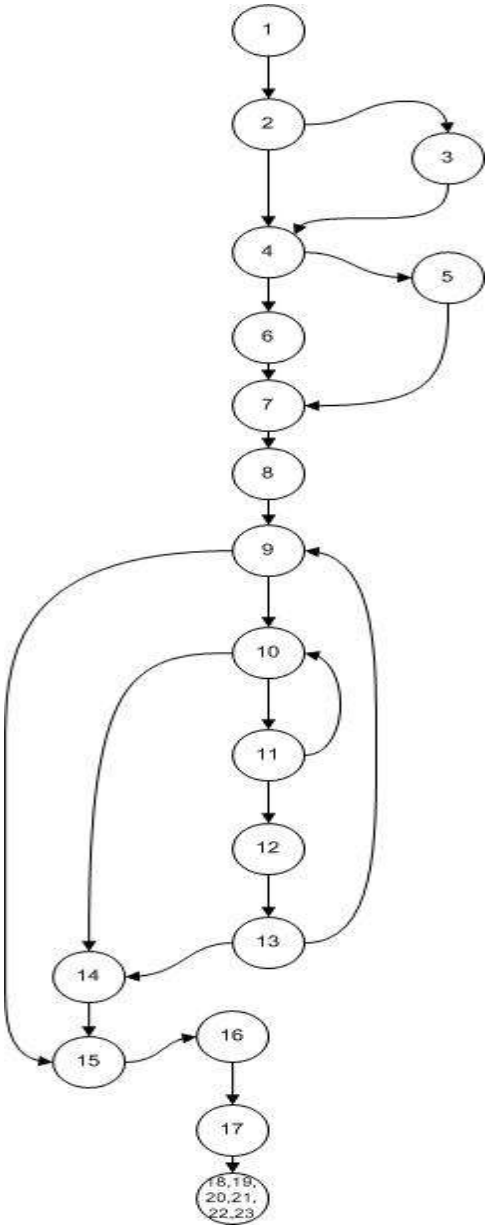
Gambar 4.5 *Flowchart Frei-Chen*



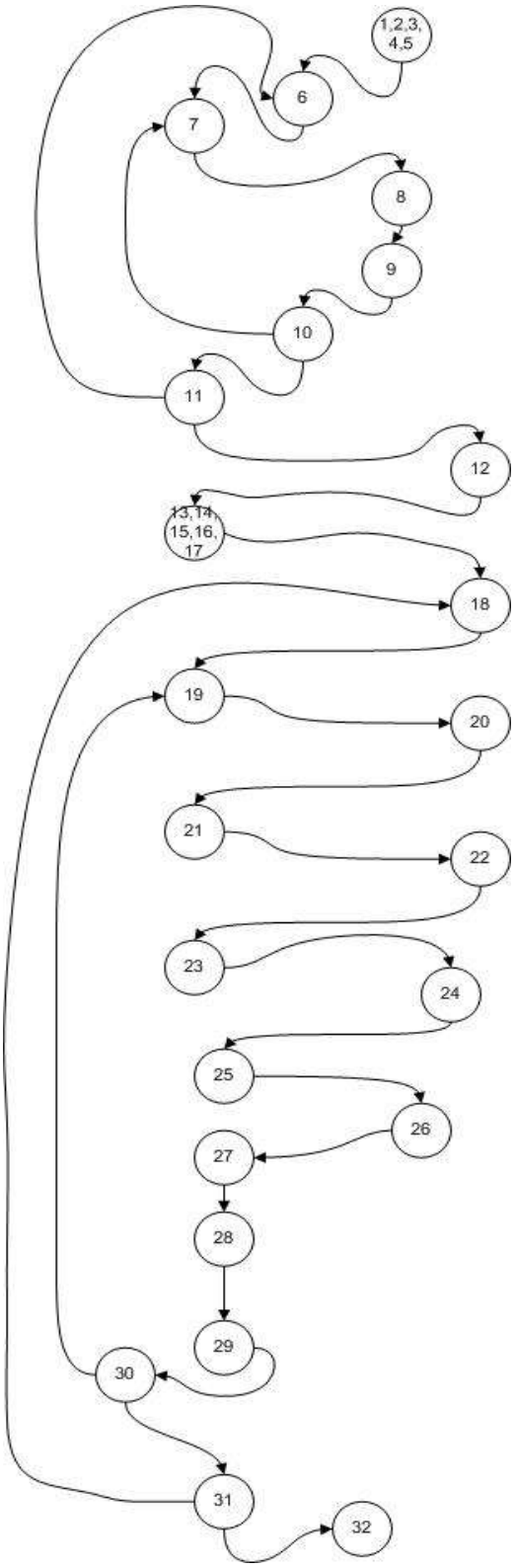
Gambar 4.6 *Flowchart SUSAN*

4.5.3 *Flow Graph Frei-Chen dan SUSAN*

Pada Gambar 4.7 dan 4.8 merupakan *flowchart* yang dimiliki pada masing-masing algoritma



Gambar 4.7 *Flow Graph Frei-Chen*



Gambar 4.8 *Flow Graph SUSAN*

4.5.4 *Cyclomatic Complexity Frei-Chen dan SUSAN*

Berikut ini adalah *cyclomatic complexity Frei-Chen*:

V(G) = E – N +2

V(G) = 23 – 18 + 2

= 7

Selanjutnya adalah *cyclomatic complexity SUSAN*:

V(G) = E – N +2

V(G) = 27 – 24 + 2

= 5

4.5.5 *Independent Path*

Berikut ini adalah *Independent Path* pada *Frei-Chen*:

Path 1: 1,2,4,6,7,8,9,15,16,17,18,19,20,21,22,23

Path 2: 1,2,4,6,7,8,9,10,14,15,16,17,18,19,20,21,22,23

Path 3: 1,2,3,4,6,7,8,9,15,16,17,18,19,20,21,22,23

Path 4: 1,2,4,5,6,7,8,9,15,16,17,18,19,20,21,22,23

Path 5: 1,2,4,6,7,8,9,10,11,10,14,15,16,17,18,19,20,21,22,23

Path 6: 1,2,4,6,7,8,9,10,11,12,13,9,15,16,17,18,19,20,21,22,23

Path 7: 1,2,4,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23

Berikut ini adalah *Independent Path* pada *SUSAN*:

Path 1: 1,2,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,
28,29,30,31,32,33

Path 2 : 1,2,4,5,6,7,8,9,10,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,
25,26,27,28,29,30,31,32,33

Path 3: 1,2,4,5,6,7,8,9,10,11,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,
23,24,25,26,27,28,29,30,31,32,33

Path 4: 1,2,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,
27,28,29,30,19,20,21,22,23,24,25,26,27,28,29,30,19,31,32,33

Path 5: 1,2,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,
27,28,29,30,31,18,19,20,21,22,23,24,25,26,27,28,29,30,19,31,32,33

4.5.6 *Graph Matrices Frei-Chen dan SUSAN*

Berikut ini adalah gambar 4.9 dan 4.10 untuk *Graph Matrices* pada masing-masinh algoritma.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
1		1																	0
2			1	1															1
3				1															0
4					1	1													1
5							1												0
6							1												0
7								1											0
8									1										0
9										1					1				1
10											1			1					1

11											1	1							1
12													1						0
13a									1					1					1
13b															1				0
13c																1			0
16																	1		0
17																		1	0
	Total																		6+1=7

Gambar 4.9 *Graph Metrices Frei-Chen*

Koneksi: 1. 1 – 1 = 0 8. 1 – 1 = 0 14. 1 – 1 = 0

2. 2 – 1 = 1 9. 2 – 1 = 1 15. 1 – 1 = 0

3.1 – 1 = 0 10. 2 – 1 = 1 16. 1 – 1 = 0

4. 2 – 1 = 1 11. 2 – 1 = 1 17. 1 – 1 = 0

5. 1 – 1 = 0 12. 1 – 1 = 0 Total komplesitas

6. 1 – 1 = 0 13. 2 – 1 = 1 *cyclomatic* = 6+1= 7

7. 1 – 1 = 0 14. 1 – 1 = 0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	
1		1																							0
2			1																						0
3				1																					0
4					1																				0
5						1																			0
6			1				1																		1
7		1						1																	1
8									1																0
9										1															0
10											1														0
11												1													0
12													1												0
13														1											0
14															1										0
15																1									0
16																	1								0
1																		1							0

7																								
18																		1						0
19																			1					0
20																				1				0
21																					1			0
22											1											1		1
23										1													1	1
																						Total		4+1 =5

Gambar 4.10 *Flow Graph SUSAN*

Koneksi: 1. 1 – 1 = 0 10. 1 – 1 = 0 19. 1 – 1 = 0

2. 1 – 1 = 0 11. 1 – 1 = 0 20. 1 – 1 = 0

3. 1 – 1 = 0 12. 1 – 1 = 0 21. 1 – 1 = 0

4. 1 – 1 = 0 13. 1 – 1 = 0 22. 1 – 1 = 0

5. 1 – 1 = 0 14. 1 – 1 = 0 23. 2 – 1 = 1

6. 2 – 1 = 1 15. 1 – 1 = 0 24. 2 – 1 = 1

7. 2 – 1 = 1 16. 1 – 1 = 0 Total komplesitas

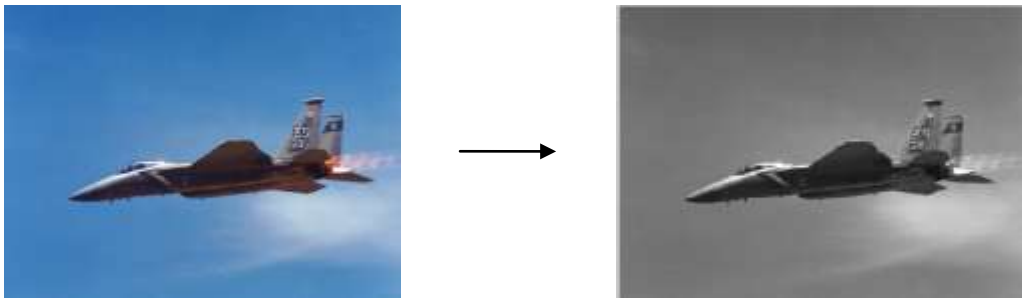
8. 1 – 1 = 0 17. 1 – 1 = 0 *cyclomatic* = 4+1 =5

9. 1 – 1 = 0 18. 1 – 1 = 0

4.6 Pengujian *Black-Box*

4.6.1 Konversi Citra Warna Menjadi Skala Keabuan

Perubahan citra warna menjadi skala keabuan, dimana memberikan intensitas pada citra warna dalam kisaran dari 0 – 255. Dimana nilai 0 merepresentasikan warna hitam sedangkan 255 merepresentasikan warna putih. Dibawah ini gambar citra warna yang diubah kedalam skala keabuan.



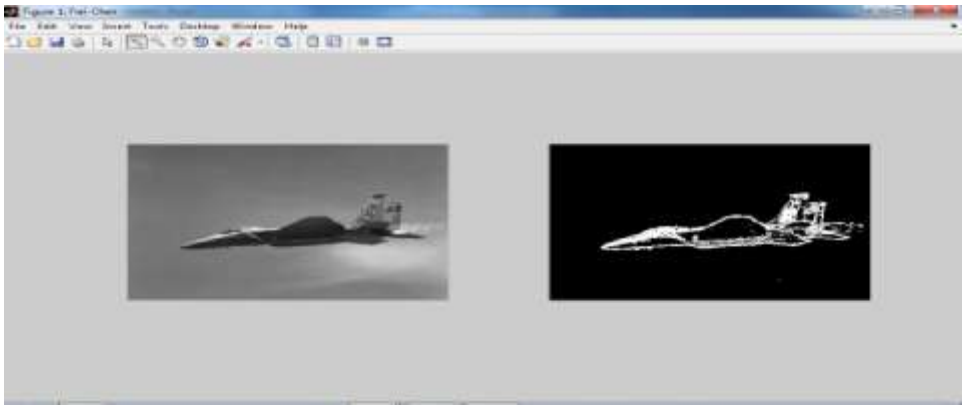
Gambar 4.11 *Perubahan Citra Warna Kedalam Skala Keabuan*

4.6.2 Deteksi Tepi Citra

Berikut ini gambar hasil dari proses deteksi tepi pada citra dengan masing-masing algoritma.

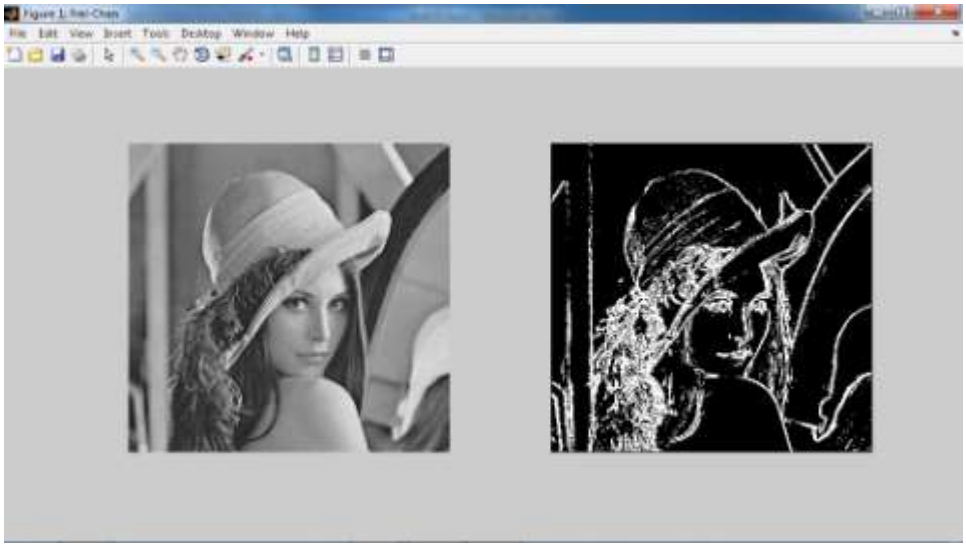
Pertama deteksi tepi dengan menggunakan algoritma frei-chen:

Citra f15.jpg dengan resolusi citra 400 x 273 piksel.



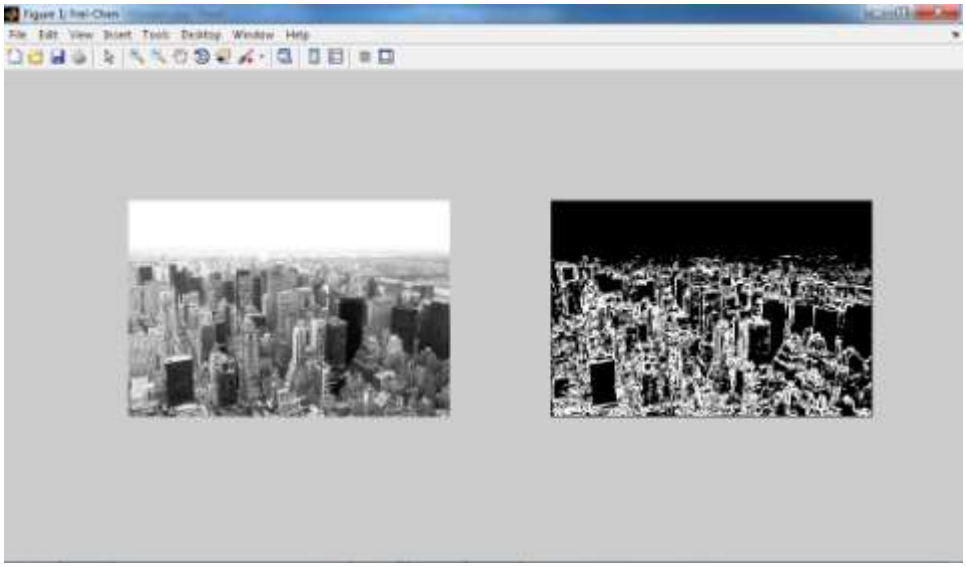
Gambar 4.12 *Hasil Deteksi Tepi Frei-Chen Pada Citra Pesawat*

Citra lena_rgb.jpg dengan resolusi 512 x 512 piksel.



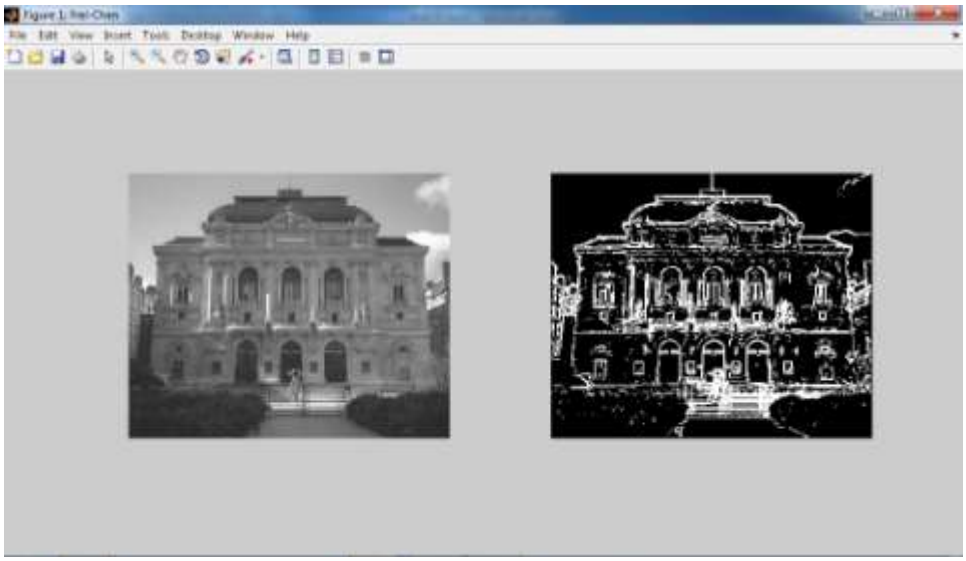
Gambar 4.13 *Hasil Deteksi Tepi Frei-Chen Pada Citra Lena*

Citra newyork.jpg dengan resolusi 400 x 271 piksel



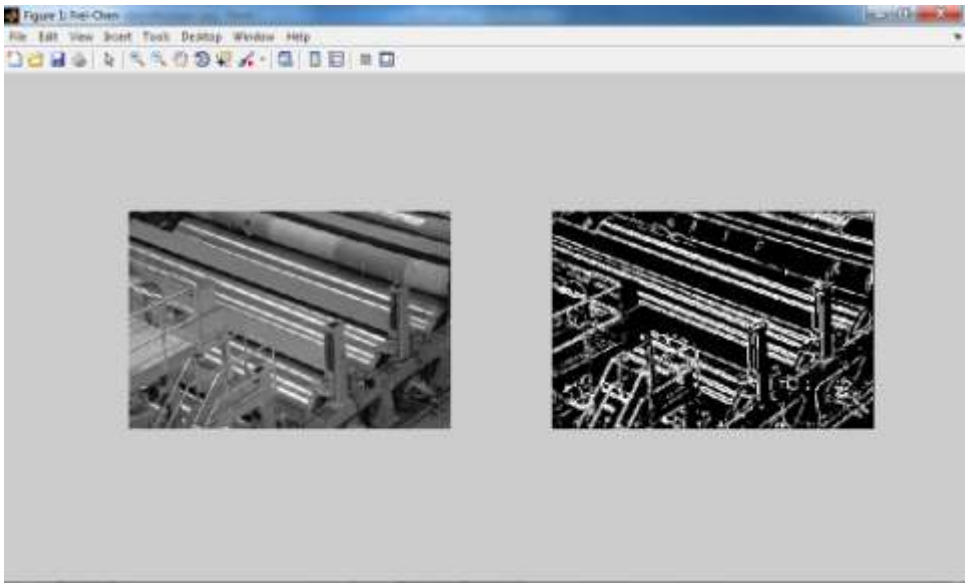
Gambar 4.14 *Hasil Deteksi Tepi Frei-Chen Pada Citra Penuh Gedung*

Citra opera.jpg dengan resolusi 400 x 337 piksel.



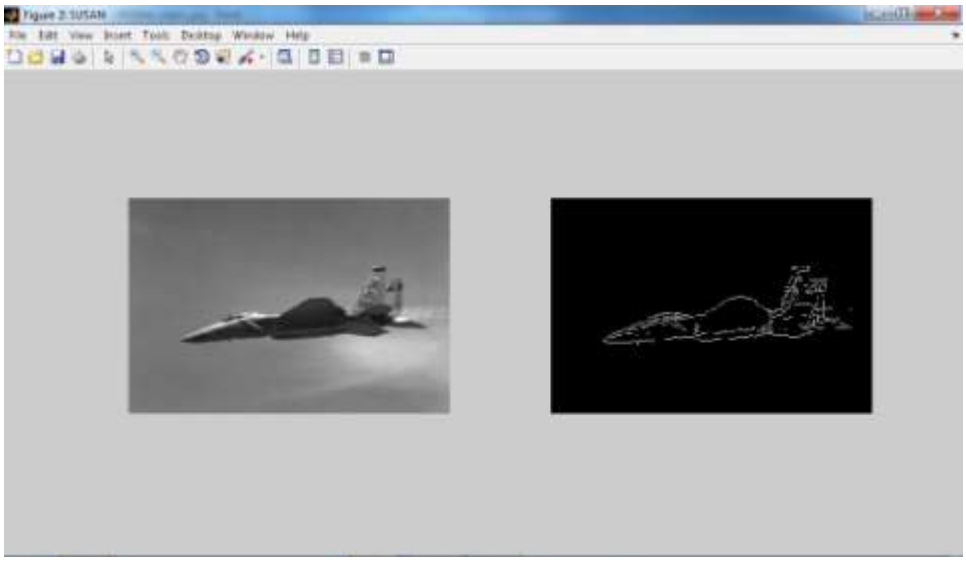
Gambar 4.15 *Hasil Deteksi Tepi Frei-Chen Pada Citra Sebuah Gedung*

Citra papermachine.jpg dengan resolusi 400 x 264 piksel

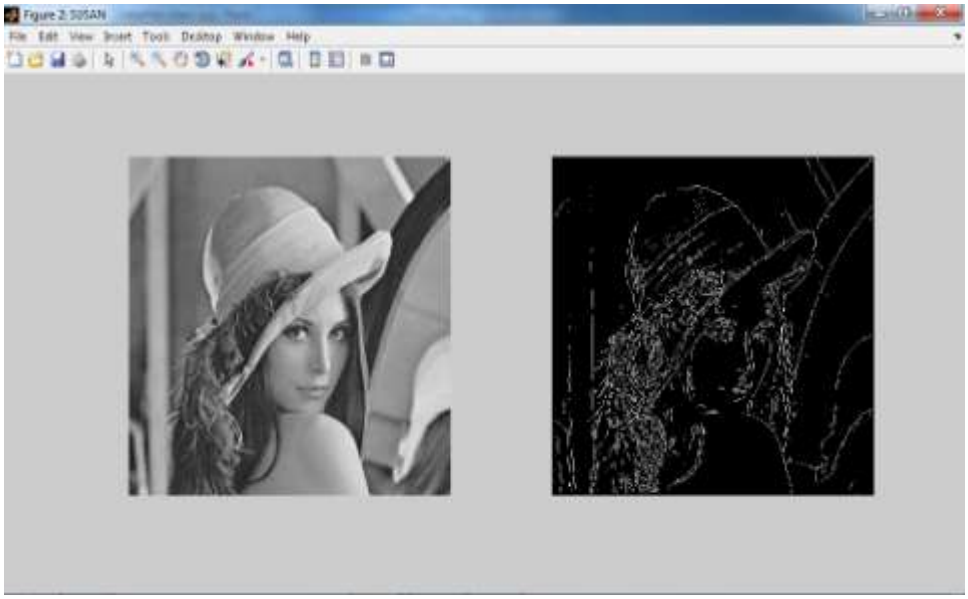


Gambar 4.16 *Hasil Deteksi Tepi Frei-Chen Pada Citra Mesin Kertas*

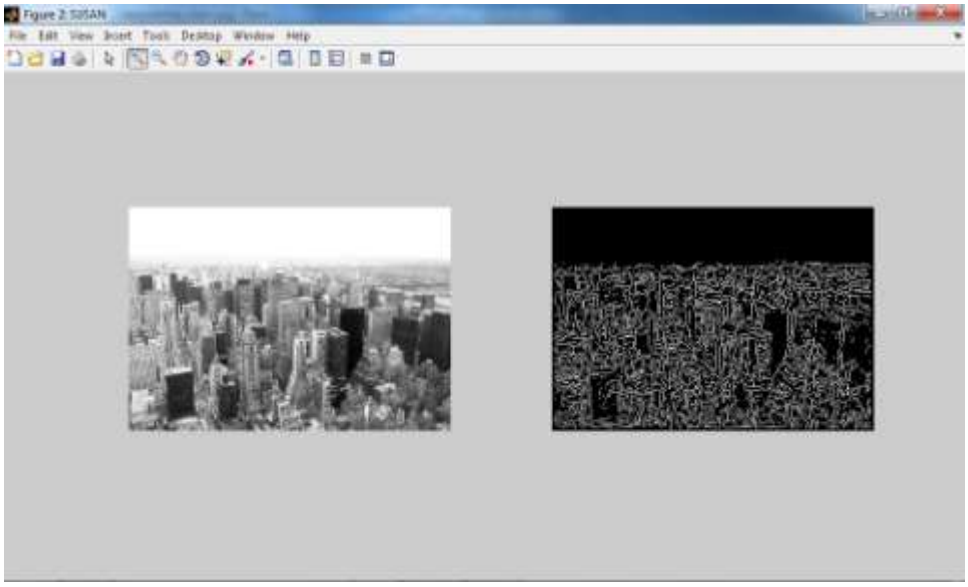
Gambar kedua dengan algoritma susan:



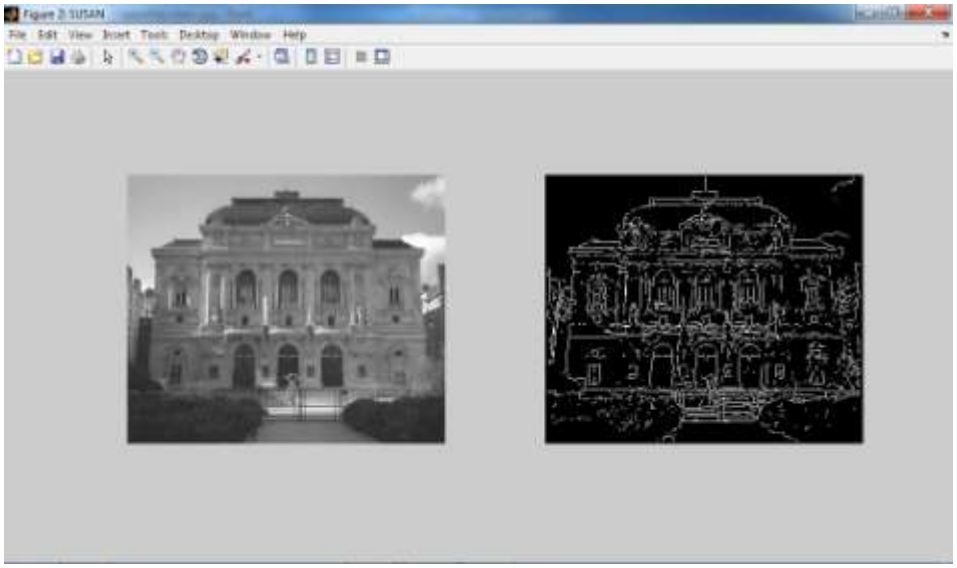
Gambar 4.17 *Hasil Deteksi Tepi SUSAN Pada Citra Pesawat*



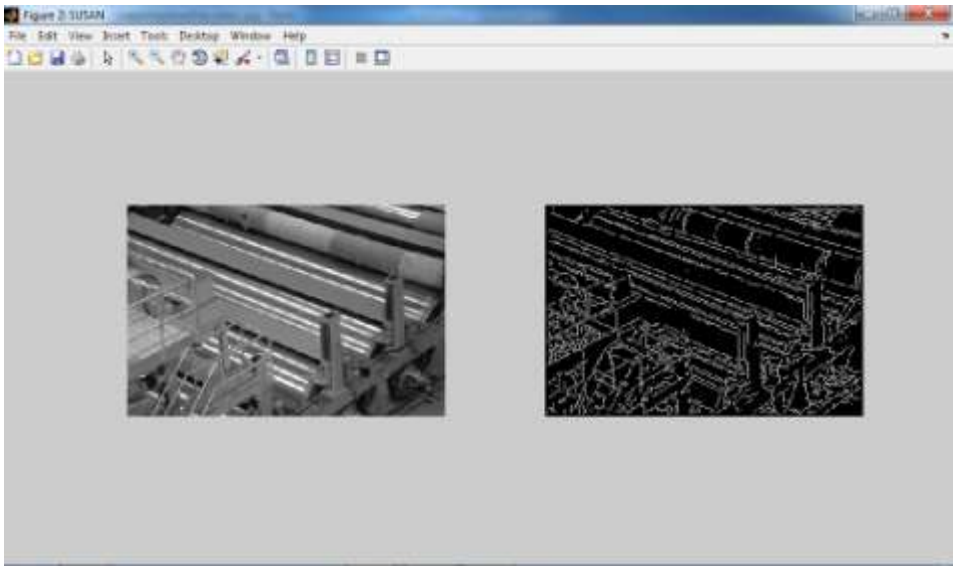
Gambar 4.18 *Hasil Deteksi Tepi SUSAN Pada Citra Lena*



Gambar 4.19 *Hasil Deteksi Tepi SUSAN Pada Citra Penuh Gedung*



Gambar 4.20 *Hasil Deteksi Tepi SUSAN Pada Citra Sebuah Gedung*



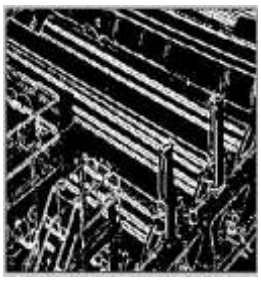
Gambar 4.21 *Hasil Deteksi Tepi SUSAN Pada Citra Mesin Kertas*

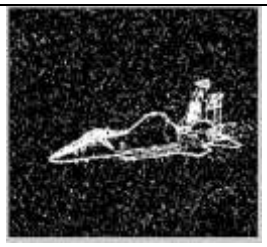
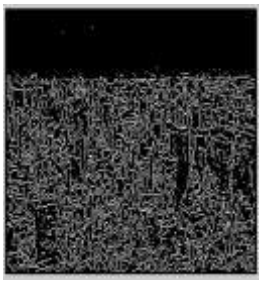
4.6.3 Deteksi Tepi Pada Citra Yang Memiliki *Noise*

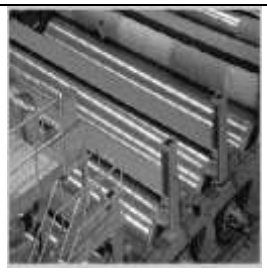
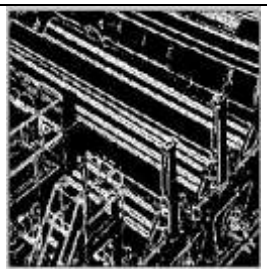
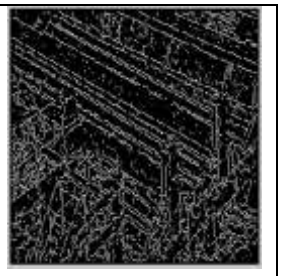
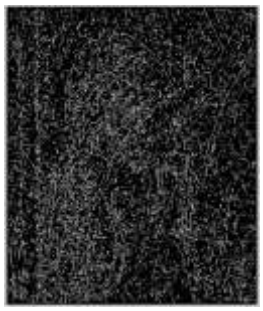
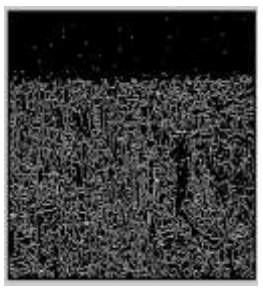
Pendeteksian tepi dengan citra yang memiliki noise, bertujuan untuk mengetahui kemampuan dari kedua algoritma dalam mendeteksi objek yang ada. Citra asli diberi *noise Gausian* dengan batasan nilai *variance* dari 0,001 – 0,005.

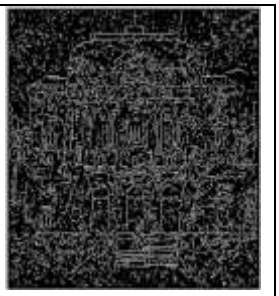
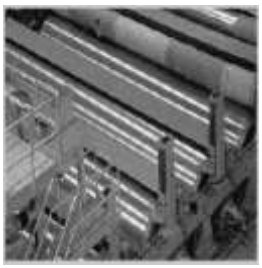
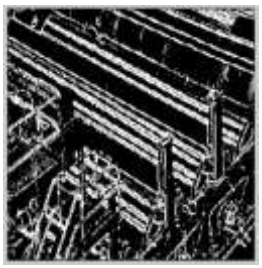
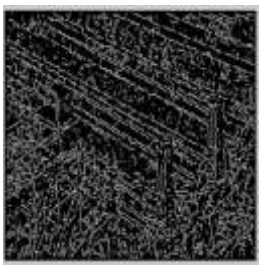
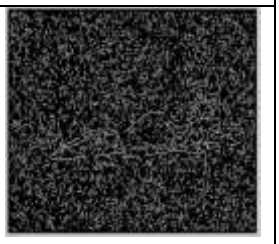
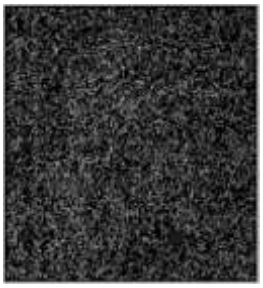
Tabel 4.1 *Hasil Deteksi Tepi Dengan Citra Bernoise*

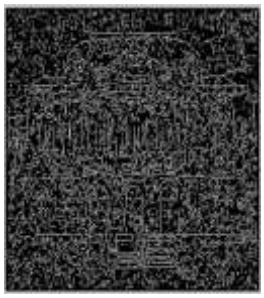
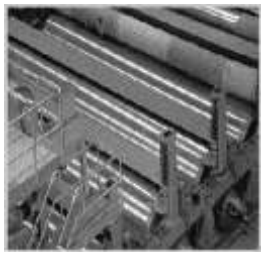
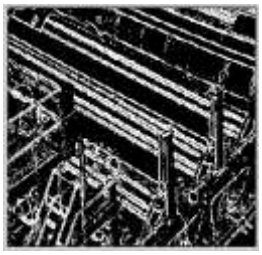
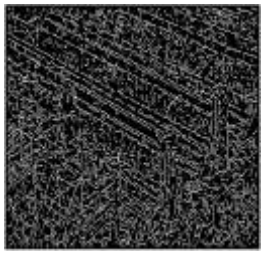
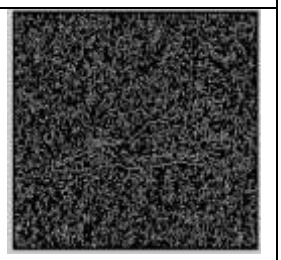
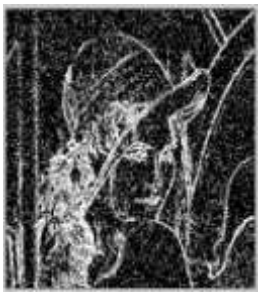
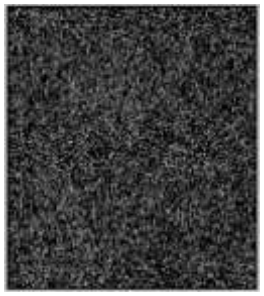
No	Nilai Variance Pada *Gausssian*	Citra Asli	Hasil Deteksi Tepi Frei-Chen	Hasil Deteksi Tepi SUSAN
1	0,001			

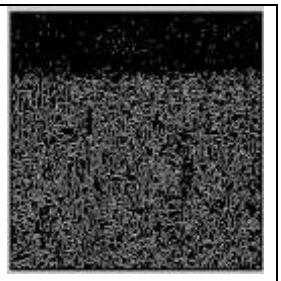
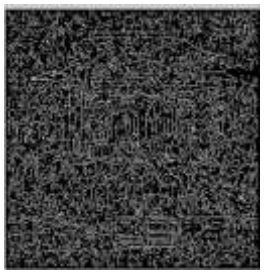
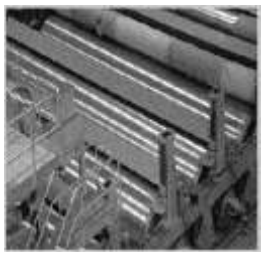
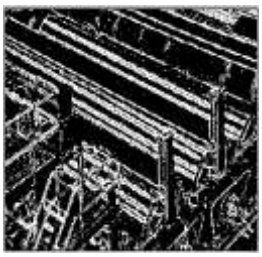
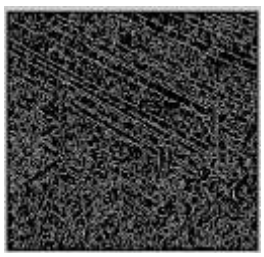
				
				
				
				

2	0,002			
				
				
				

				
3	0,003			
				
				

				
				
4	0,004			
				
				

				
				
5	0,005			
				

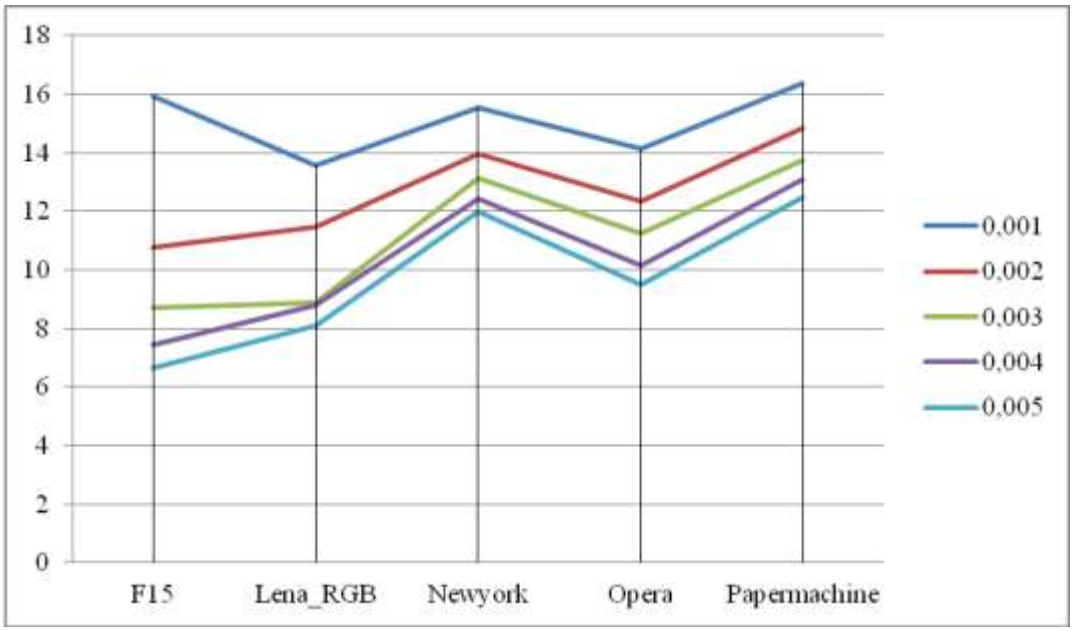
4.6.4 Perbandingan Deteksi Tepi

Perbandingan deteksi tepi merupakan berdasarkan perbandingan waktu proses dan nilai PSNR dari kedua metode yang digunakan. Berikut tabel hasil dari waktu proses, nilai MSE dan nilai PSNR yang diperoleh.

Tabel berikut memperlihatkan pendeteksian tepi *frei-chen* pada citra *bernoise.*

Tabel 4.2 *Perhitungan PSNR Frei-Chen*

Nilai *variance*	F15	Lena_RGB	Newyork	Opera	Papermachine
0,001	15,9262	13,5773	15,5445	14,1344	16,3699
0,002	10,7524	11,4788	13,9624	12,3416	14,8258
0,003	8,69136	8,8966	13,1338	11,249	13,7307
0,004	7,46011	8,79276	12,4159	10,1668	13,0716
0,005	6,64704	8,08999	11,9776	9,49363	12,4754

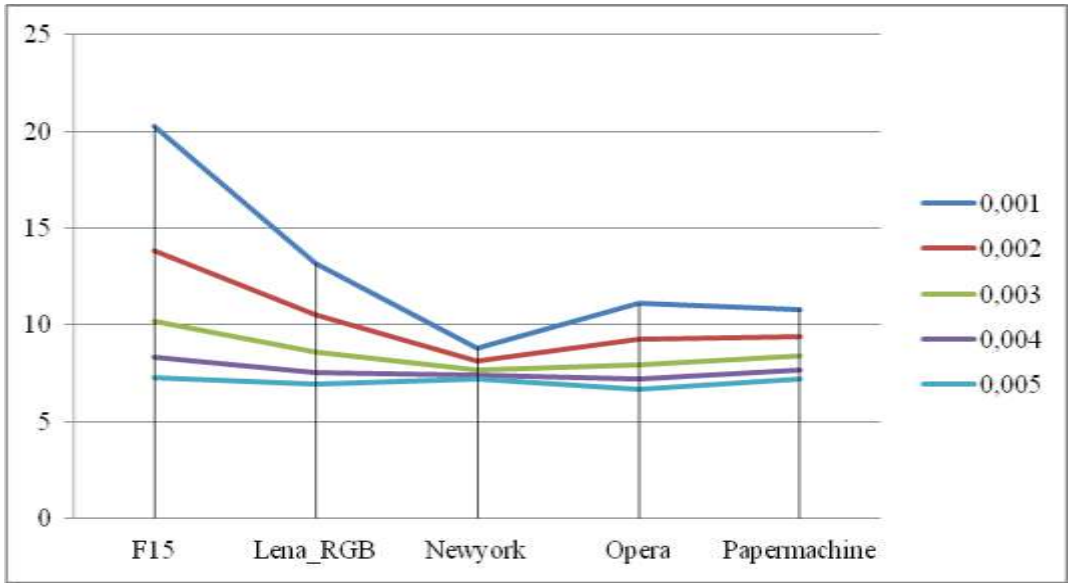


Gambar 4.22 *Grafik PSNR Untuk Deteksi Tepi Frei-Chen*

Tabel berikut memperlihatkan pendeteksian tepi *SUSAN* pada citra *bernoise*.

Tabel 4.3 *Perhitungan PSNR SUSAN*

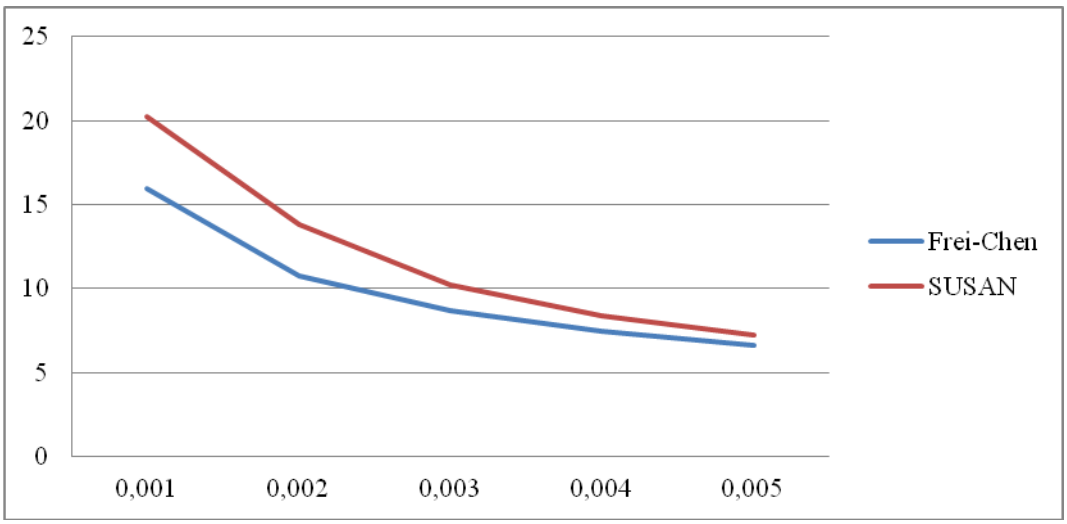
Nilai *variance*	F15	Lena_RGB	Newyork	Opera	Papermachine
0,001	20,2581	13,2028	8,81519	11,1178	10,7938
0,002	13,8203	10,5245	8,13667	9,29798	9,41921
0,003	10,204	8,60251	7,67177	7,95357	8,38547
0,004	8,35571	7,55518	7,4096	7,18491	7,65961
0,005	7,26426	6,91968	7,18744	6,68146	7,18614



Gambar 4.23 *Grafik PSNR Untuk Deteksi Tepi SUSAN*

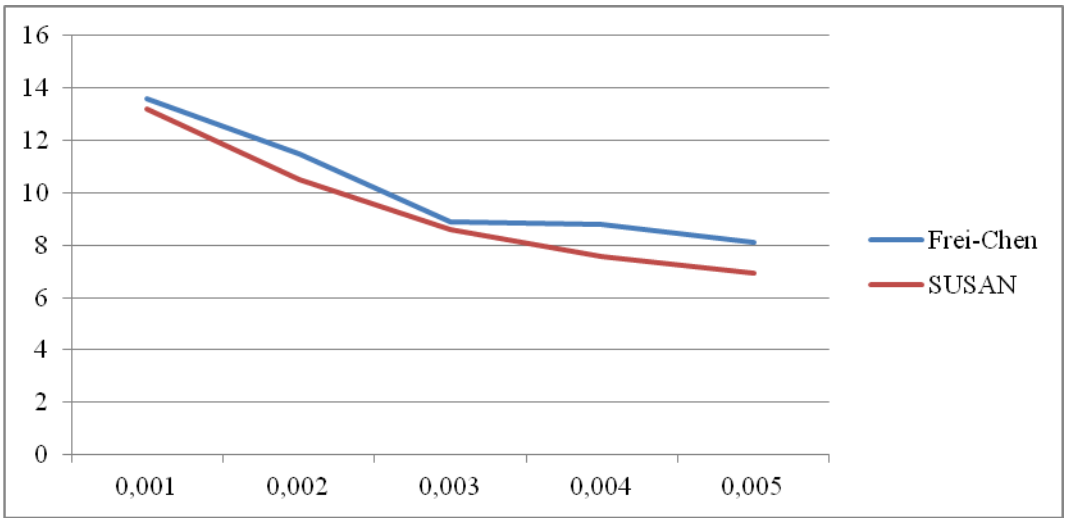
Dari data-data diatas maka diperoleh hasil perbandingan antara kedua metode.

Berikut ini pada gambar 4.16 menampilkan perbandingan nilai *PSNR* pada citra *F15.*



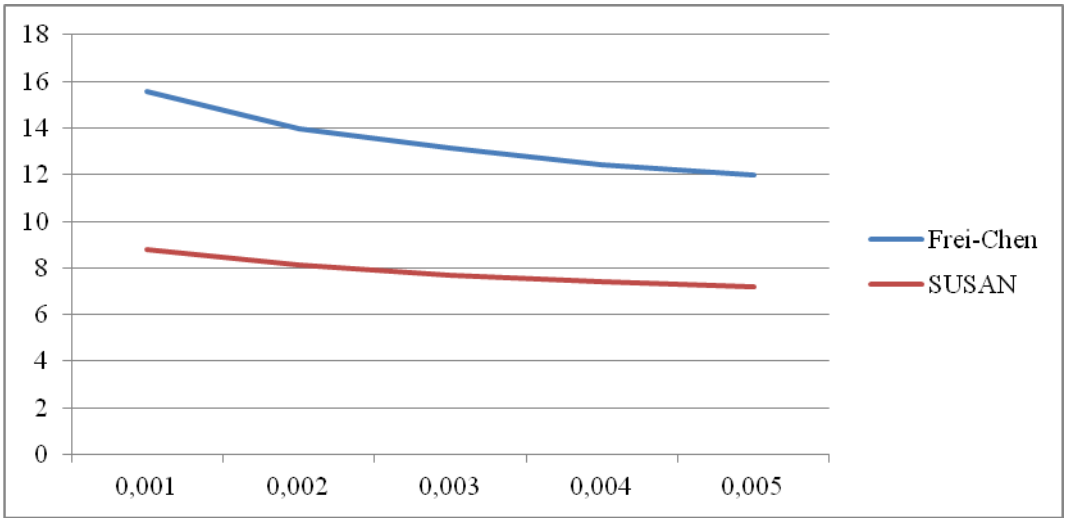
Gambar 4.24 *Grafik perbandingan nilai PSNR Untuk citra F15*

Berikut ini pada gambar 4.17 menampilkan perbandingan nilai *PSNR* pada citra *Lena_RGB.*



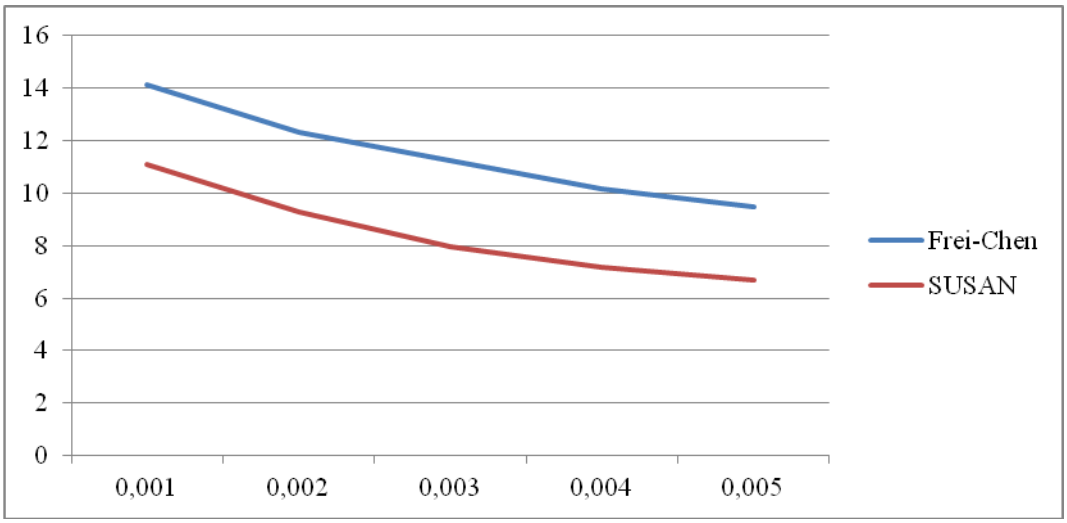
Gambar 4.25 *Grafik perbandingan nilai PSNR Untuk citra Lena_RGB*

Berikut ini pada gambar 4.18 menampilkan perbandingan nilai *PSNR* pada citra *Newyork.*



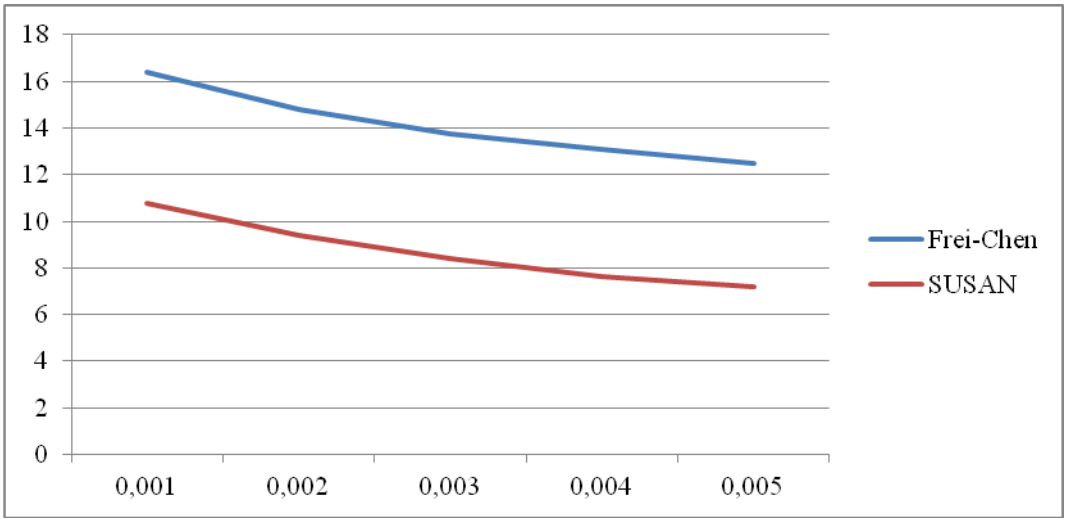
Gambar 4.26 *Grafik perbandingan nilai PSNR Untuk citra Newyork*

Berikut ini pada gambar 4.19 menampilkan perbandingan nilai *PSNR* pada citra *Opera.*



Gambar 4.27 *Grafik perbandingan nilai PSNR Untuk citra Opera*

Berikut ini pada gambar 4.20 menampilkan perbandingan nilai *PSNR* pada citra *Papermachine.*



Gambar 4.28 *Grafik perbandingan nilai PSNR Untuk citra Papermachine*

4.6.5 Pembahasan Hasil Penelitian

Dari hasil penelitian yang dilakukan dari perbandingan dua algoritma deteksi tepi yang berbeda yaitu *frei-chen* dan *SUSAN* kinerja dalam mendeteksi tepi pada citra yang diberikan berhasil dengan baik.

Pada citra *F15.jpg* kedua algoritma berhasil mendeteksi tepi sebuah pesawat. Ketika gambar diberikan *noise Gaussian* dengan nilai 0,001, kedua algoritma masih memberikan hasil yang baik. Begitu pula pada saat diberikan nilai 0,002, kedua algoritma masih bisa mendeteksi tepi dengan baik. Pada saat pemberian nilai 0,003 algoritma susan mulai menghasilkan deteksi tepi yang tidak sesuai, karena banyaknya noise diberikan, sedangkan algoritma frei-chen masih bisa mendeteksi tepi pada objek pesawat, meskipun *noise* yang diberikan dideteksi juga. Pada nilai 0,004 sampai 0,005 algoritma susan sudah tidak mampu mendeteksi tepi dengan baik, gambar yang dihasilkan titik-titik putih yang dihasilkan oleh *noise* yang diberikan, sedangkan pada algoritma frei-chen

meskipun sama tetapi objek yang di deteksi tepinya memberikan gambar garis yang kuat, jadi masih bisa dilihat hasil deteksi tepi pesawatnya. Pada gambar 4.16 perbandingan antara kedua algoritma pada nilai 0,001 sampai 0,005 algoritma *SUSAN* memperoleh nilai *PSNR* yang tinggi dibandingkan dengan algoritma *frei-chen.*

Pada citra *lena_RGB.png* pada saat pemberian nilai 0,001, deteksi tepi yang dihasilkan masih baik. Tetapi pada saat diberikan nilai 0,002, algoritma susan mulai tidak mampu mendeteksi tepi dengan baik terlihat dari banyaknya *noise* yang terdeteksi sehingga mempengaruhi pendeteksian tepi. Sehingga sampai pada nilai 0,005 algoritma susan sudah tidak berhasil mendeteksi tepi pada citra *lena.jpg*. Pada gambar 4.17 perbandingan terlihat *frei-chen* mempunyai nilai *PSNR* yang lebih baik dibandingkan dengan algoritma *SUSAN*, dari nilai 0,001 sampai 0,005.

Pada citra *newyork.jpg* memberikan hasil yang sama baik pada *noise* yang bernilai 0,001, tetapi pada algoritma susan gedung-gedung yang ada tidak bisa dikenali walaupun berhasil dideteksi tepinya. Sehingga sampai *noise* yang bernilai 0,005 algoritma susan memberikan deteksi tepi yang objeknya tidak bisa dikenali. Sedangkan pada algoritma frei-chen gedung-gedung berhasil dideteksi tepinya dan objeknya dikenali dengan baik. Pada gambar 4.18 algoritma *frei-chen* mendapatkan nilai *PSNR* yang lebih baik lagi dibandingkan dengan algoritma *SUSAN.*

Pada citra *opera.jpg* memberikan hasil yang mirip dengan citra *lena_RGB.jpg*, dimana pada saat diberikan nilai 0,002, algoritma susan sudah sulit

mendeteksi tepi gedung. Begitu pula pada saat diberikan nilai 0,005. Pada gambar 4.19 algoritma *frei-chen* mendapatkan nilai yang lebih tinggi dibandingkan dengan algoritma *SUSAN*.

Pada citra *papermachine.png* memberikan hasil yang mirip dengan *newyork.jpg*. algoritma susan mulai sulit mendeteksi tepi pada objek yang sudah diberikan noise yang bernilai 0,002 sampai dengan 0,005. Pada gambar 4.20 algoritma *frei-chen* mendapatkan nilai *PSNR* yang lebih baik dibandingkan dengan algoritma *SUSAN*.

Kemudian dari hasil nilai *PSNR* pada masing-masing algoritma, didapatkan bahwa algoritma susan memiliki nilai *PSNR* yang lebih baik pada citra yang diberi nilai *noise* sebesar 0,001. Sedangkan pada algoritma frei-chen nilai *PSNR* yang lebih baik pada nilai *noise* dari 0,002 sampai 0,005. Jadi algoritma *frei-chen* memberikan nilai *PSNR* terbaik rata-rata, dibandingkan dengan algoritma *SUSAN*.