

**LAPORAN *PROJECT***  
**PENGOLAHAN CITRA**



**Disusun oleh :**

**Nama:** Fathan Arsyadani

**NIM :** 4611418047

**Prodi :** Teknik Informatika

**UNIVERSITAS NEGERI SEMARANG**

**2019/2020**

# DAFTAR ISI

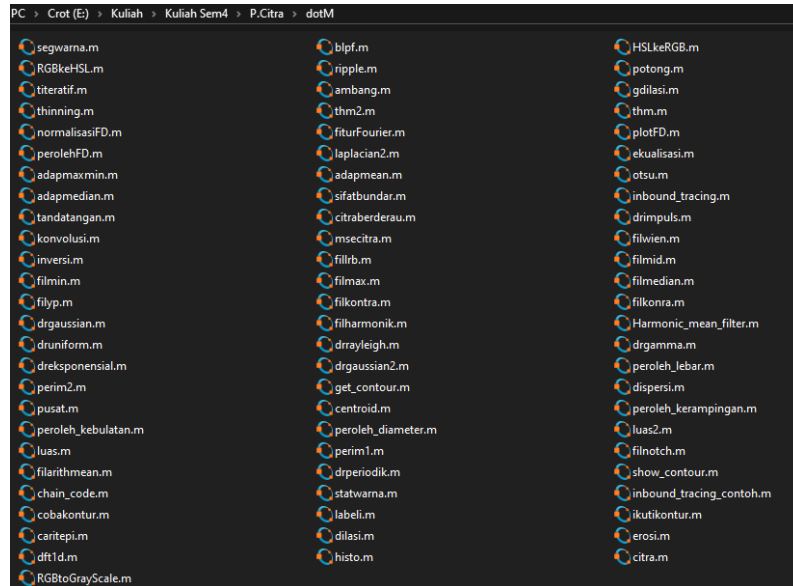
<b>HALAMAN SAMPUL .....</b>	<b>i</b>
<b>DAFTAR ISI.....</b>	<b>ii</b>
<b>A. Alat dan Bahan .....</b>	<b>1</b>
a. Alat .....	1
b. Bahan .....	1
<b>B. Praktikum .....</b>	<b>2</b>
1. <i>Load</i> citra ke octave .....	2
a. <i>Command</i> .....	2
2. Ekstraksi tepi objek ( <i>caritepi.m</i> ) .....	2
a. Kode Program.....	3
b. Pemanggilan <i>function caritepi.m</i> .....	3
c. Hasil .....	3
3. Penambahan/Pembangkitan derau eksponensial ( <i>dreksponensial.m</i> ) .....	3
a. Kode Program.....	4
b. Pemanggilan <i>function dreksponensial.m</i> .....	4
c. Hasil .....	4
4. Deteksi tepi objek menggunakan konvolusi ( <i>konvolusi.m</i> ).....	4
a. Kode Program.....	5
b. Pemanggilan <i>function konvolusi.m</i> .....	5
c. Hasil .....	5
5. Deteksi tepi menggunakan operator <i>laplacian</i> ( <i>laplacian2.m</i> ).....	5
a. Kode Program.....	5
b. Pemanggilan <i>function laplacian2.m</i> .....	6
c. Hasil .....	6
6. Mendapatkan kontur menggunakan deskriptor <i>fourier</i> ( <i>perolehFD.m</i> dan <i>plotFD.m</i> ) .....	6
a. Kode Program <i>perolehFD.m</i> .....	6
b. Pemanggilan <i>function perolehFD.m</i> (sebelumnya telah dilakukan <i>inbound_tracing</i> terhadap citra biner pika).....	6
c. Kode Program <i>plotFD.m</i> .....	7
d. Pemanggilan <i>function plotFD.m</i> .....	7
e. Hasil .....	7

<b>7. Melakukan pengambangan dwi-aras pada citra(ambang.m dan titeratif.m)</b> .....	7
a. Kode Program <b>titeratif.m</b> .....	8
b. Pemanggilan <i>function</i> <b>titeratif.m</b> .....	8
c. Kode Program <b>ambang.m</b> .....	8
d. Pemanggilan <i>function</i> <b>ambang.m</b> .....	8
e. Hasil <i>function</i> <b>ambang.m</b> .....	9
<b>8. Melakukan pemotongan aras keabuan (potong.m)</b> .....	9
a. Kode Program .....	9
b. Pemanggilan <i>function</i> <b>potong.m</b> .....	9
c. Hasil .....	10
<b>9. Memberikan efek bergelombang pada citra (ripple.m)</b> .....	10
a. Kode Program .....	10
b. Pemanggilan <i>function</i> <b>ripple.m</b> .....	10
c. Hasil .....	11
<b>10. Membuat citra menjadi blur menggunakan Butterworth Low Pass Filter (blpf.m)</b> .....	11
a. Kode Program .....	11
b. Pemanggilan <i>function</i> <b>blpf.m</b> .....	11
c. Hasil .....	12
<b>11. Melakukan segmentasi warna pada citra (segwarna.m)</b> .....	12
a. Kode Program .....	12
b. Pemanggilan <i>function</i> <b>segwarna.m</b> .....	12
c. Hasil .....	13

## A. Alat dan Bahan

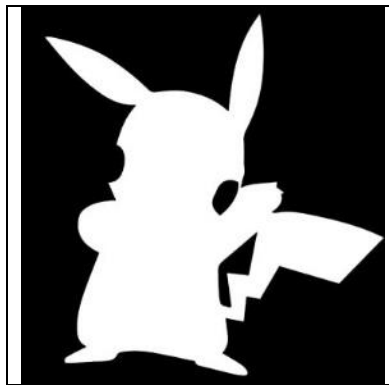
### a. Alat

- Octave
- Beberapa *function*



### b. Bahan

- Citra pika.jpg



Citra pika.jpg

- Citra cici.jpg



Citra cici.jpg

- Citra danilla.jpg



Citra danilla.jpg

## B. Praktikum

### 1. Load citra ke octave

#### a. Command

```
>> img=imread('E:\Kuliah\Kuliah Sem4\P.Citra\bahan\binerpika.jpg');
>> biner=im2bw(img);
```

Citra pika.jpg

```
>> abuabu=imread('E:\Kuliah\Kuliah Sem4\P.Citra\bahan\cici.jpg');
```

Citra cici.jpg

```
>> berwarna=imread('E:\Kuliah\Kuliah Sem4\P.Citra\bahan\danilla.jpg');
```

Citra danilla.jpg

### 2. Ekstraksi tepi objek (caritepi.m)

Tepi objek pada citra biner dapat diperoleh melalui algoritma yang dibahas oleh Davis (1990). Pemrosesan dilakukan dengan menggunakan 8-ketetanggaan. Algoritma perolehan tepi ini mengasumsikan bahwa semua piksel pada kolom pertama dan terakhir serta baris pertama dan baris terakhir tidak ada yang bernilai 1. **Perolehan tepi objek ini dapat dilakukan menggunakan *function caritepi.m*.**

a. Kode Program

```
function [G]=caritepi(F)
[jum_baris,jum_kolom]=size(F);
G=zeros(jum_baris,jum_kolom);

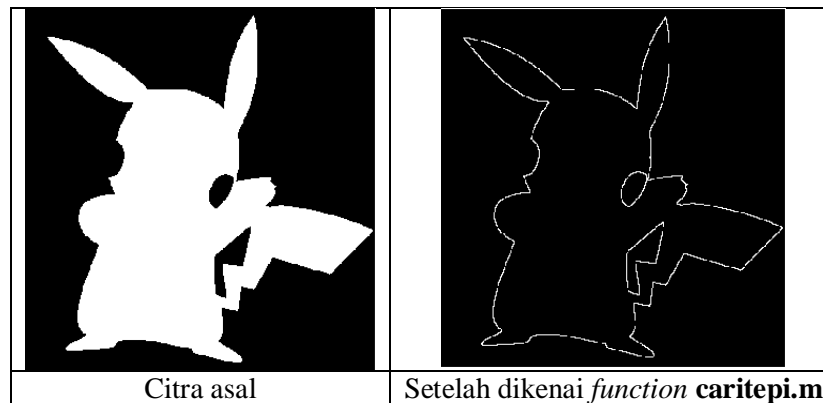
for q=2:jum_baris-1
    for p=2:jum_kolom-1
        p0=F(q,p+1);
        p1=F(q-1,p+1);
        p2=F(q-1,p);
        p3=F(q-1,p-1);
        p4=F(q,p-1);
        p5=F(q+1,p-1);
        p6=F(q+1,p);
        p7=F(q+1,p+1);
        sigma=p0+p1+p2+p3+p4+p5+p6+p7;
        if sigma==3
            G(q,p)=0;
        else
            G(q,p)=F(q,p);
        endif
    endfor
endfor
endfunction
```

caritepi.m

b. Pemanggilan *function caritepi.m*

```
>> A=caritepi(biner);
>> imshow(A)
```

c. Hasil



### 3. Penambahan/Pembangkitan derau eksponensial (dreksponensial.m)

Derau Eksponensial (terkadang dinamakan derau eksponensial negatif) merupakan jenis derau yang dihasilkan oleh laser yang koheren ketika ctra diperoleh. Oleh karena itu derau ini sering disebut sebagai bercak laser (Myler and Weeks,1993). **Pembangkitan derau eksponensial dapat dilakukan dengan *function dreksponensial.m*.**

a. Kode Program

```
function [G]=dreksponensial(F,a)

if nargin~=2
    error('Penggunaan: dreksponensial(F,a)');
endif

if a<=0
    error('Parameter berupa sebarang bilangan>0');
endif

[m,n]=size(F);

F=double(F);
G=zeros(m,n);
for i=1:m
    for j=1:n
        derau=-1/a*log(1-rand);

        G(i,j)=round(F(i,j)+derau);
        if G(i,j)>255
            G(i,j)=255;
        endif
    endfor
endfor

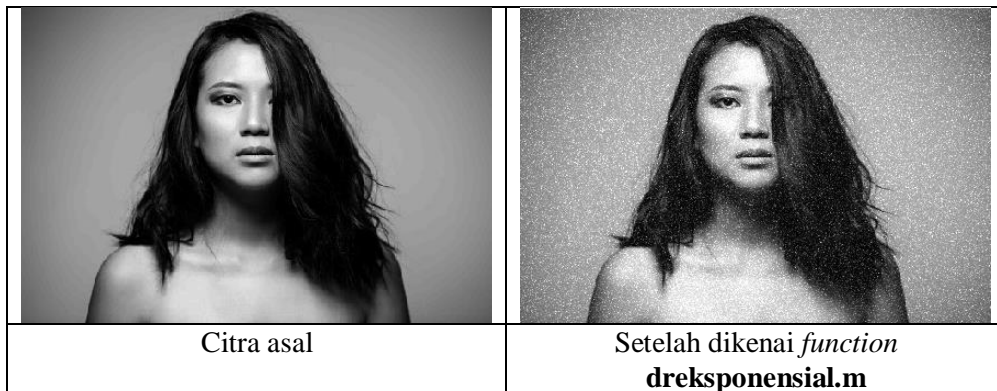
G=uint8(G);
endfunction
```

dreksponensial.m

b. Pemanggilan *function* **dreksponensial.m**

```
>> B=dreksponensial(abuabu,0.05);
```

c. Hasil



#### 4. Deteksi tepi objek menggunakan konvolusi (konvolusi.m)

Konvolusi seringkali dilibatkan dalam operasi ketetanggaan piksel. Konvolusi pada citra didefinisikan sebagai proses untuk memperoleh suatu piksel didasarkan pada nilai piksel itu sendiri dan piksel tetangganya, dengan melibatkan suatu matriks yang disebut kernel yang merepresentasikan pembobotan. Pada pelaksanaan konvolusi, kernel digeser sepanjang baris dan kolom dalam citra sehingga diperoleh nilai yang baru pada citra keluaran. **Operasi konvolusi pada citra dapat dilakukan dengan menggunakan *function* konvolusi.m.**

a. Kode Program

```
function [G]=konvolusi(F,H)
[tinggi_f,lebar_f]=size(F);
[tinggi_h,lebar_h]=size(H);

m2=floor(tinggi_h/2);
n2=floor(lebar_h/2);


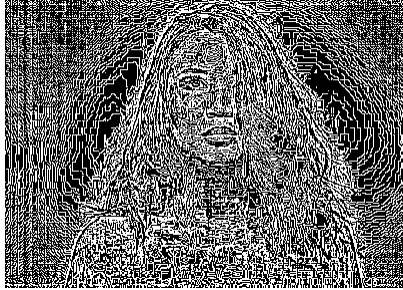

F2=double(F);
for y=m2+1:tinggi_f-m2
    for x=n2+1:lebar_f-n2
        jum=0;
        for p=-m2:m2
            for q=-n2:n2
                jum=jum+H(p+m2+1,q+n2+1)*F2(y-p,x-q);
            endfor
        endfor
        G(y-m2,x-n2)=jum;
    endfor
endfor
endfunction
```

konvolusi.m

b. Pemanggilan *function* **konvolusi.m**

```
H=[-1 0 -1;0 4 0;-1 0 -1];
C=konvolusi(abuabu,H);
C2=uint8(C);
```

c. Hasil

		
Citra asal	Setelah dikenai <i>function</i> <b>konvolusi.m</b>	Citra hasil konvolusi setelah dilakukan pengaturan nilai piksel agar berada pada jangkauan [0-255] menggunakan <i>function</i> <b>uint8</b>

## 5. Deteksi tepi menggunakan operator *laplacian* (laplacian2.m)

Operator *Laplacian* merupakan contoh operator yang berdasarkan pada turunan kedua. Operator ini bersifat *omnidirectional*, yakni menebalkan bagian tepi ke segala arah. **Pengujian operator *Laplacian* terhadap citra berskala keabuan dapat dilakukan dengan menggunakan *function* laplacian2.m.**

a. Kode Program

```
function [G]=laplacian2(F)

[m,n]=size(F);
G=zeros(m,n);

F=double(F);
for y=2:m-1
    for x=2:n-1
        G(y,x)=8*(F(y,x)-(F(y-1,x)+F(y,x-1)+F(y,x+1)+F(y+1,x)+...
            F(y-1,x-1)+F(y-1,x+1)+F(y+1,x-1)+F(y+1,x+1)));
    endfor
endfor
G=uint8(G);
endfunction
```

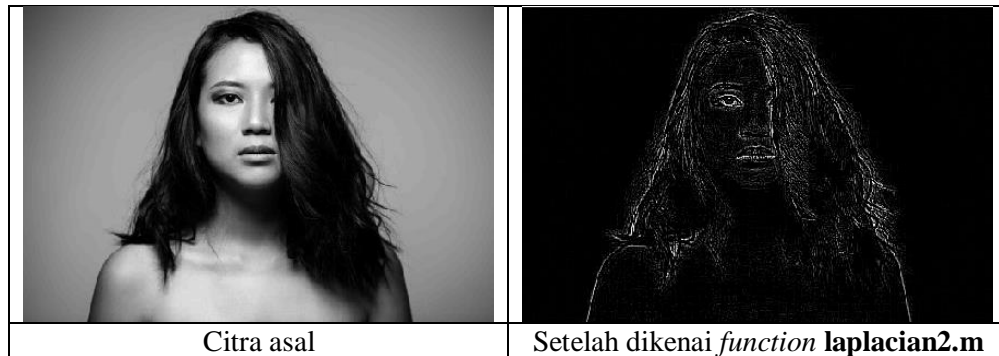
laplacian2.m



- b. Pemanggilan *function* **laplacian2.m**

```
>> D=laplacian2(abuabu);
```

- c. Hasil



## 6. Mendapatkan kontur menggunakan deskriptor *fourier*(perolehFD.m dan plotFD.m)

Deskriptor *Fourier* biasa dipakai untuk menjabarkan bentuk dalam dua dimensi dengan menggunakan transformasi *Fourier*. Dengan menggunakan deskriptor *Fourier*, suatu bentuk dapat dinyatakan dengan sejumlah bilangan (yaitu koefisien *Fourier*). Konsep dasar untuk mendapatkan deskriptor *Fourier* adalah dengan mendapatkan kontur objek terlebih dahulu kemudian piksel-piksel di kontur ditransformasikan menggunakan FFT. **Hal ini dapat dilakukan oleh *function* perolehFD.m.**

- a. Kode Program perolehFD.m

```
function [F]=perolehFD(Kontur)

jum=length(Kontur);

if rem(jum,2)==1
    Kontur=[Kontur;Kontur(1,:)];
endif

K=Kontur(:,2)-i*Kontur(:,1);
F=fft(K);
endfunction
```

**perolehFD.m**

- b. Pemanggilan *function* **perolehFD.m** (sebelumnya telah dilakukan *inbound\_tracing* terhadap citra biner pika).

```
>> I=inbound_tracing(biner);
>> J=perolehFD(I);
```

Dalam hal ini, J berisi koefisien-koefisien *Fourier* dari citra biner pika. Koefisien-koefisien yang tercatat pada J dapat digunakan untuk **membuat kontur objek menggunakan *function* plotFD.m.**

c. Kode Program **plotFD.m**

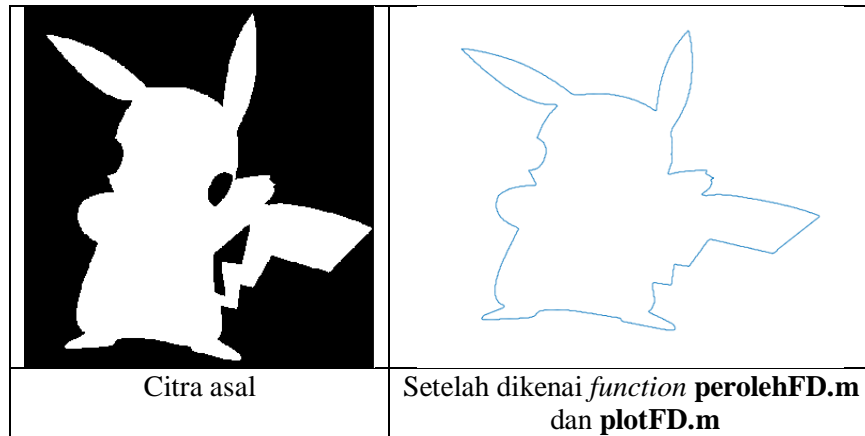
```
function []=plotFD(F)
    jum=length(F);
    if jum>0
        G=ifft(F);
        G=[G; G(1)];
        plot(G);
        axis off;
    endif
endfunction
```

**plotFD.m**

d. Pemanggilan *function* **plotFD.m**

```
>> plotFD(J)
```

e. Hasil



## 7. Melakukan pengambangan dwi-aras pada citra(ambang.m dan titeratif.m)

Pengambangan dwi-aras adalah salah satu segmentasi citra dengan memanfaatkan ambang intensitas. Nilai yang lebih kecil daripada nilai ambang diperlakukan sebagai area pertama dan yang lebih besar atau sama dengan nilai ambang dikelompokkan sebagai area kedua. Persoalan utama dalam pengambangan dwi-aras terletak pada penentuan nilai ambang (t). Oleh karena itu, kita perlu **menentukan nilai ambang dari suatu citra terlebih dahulu dengan *function* titeratif.m.**

a. Kode Program **titeratif.m**

```
function [t1]=titeratif(F)

[m,n]=size(F);
F=double(F);
t0=127;
while true
    rata_kiri=0;
    rata_kanan=0;
    jum_kiri=0;
    jum_kanan=0;
    for i=1:m
        for j=1:n
            if F(i,j)<=127
                rata_kiri=rata_kiri+F(i,j);
                jum_kiri=jum_kiri+1;
            else
                rata_kanan=rata_kanan+F(i,j);
                jum_kanan=jum_kanan+1;
            endif
        endfor
    endfor
    rata_kiri=rata_kiri/jum_kiri;
    rata_kanan=rata_kanan/jum_kanan;
    t1=(rata_kiri+rata_kanan)/2.0;
    if (t0-t1)<1
        break;
    endif
    t0=t1;
endwhile
t1=floor(t1);
endfunction
```

**titeratif.m**

b. Pemanggilan *function* **titeratif.m**

```
>> t=titeratif(abuabu)
t = 107
```

Setelah nilai iteratif ditentukan, barulah **pengembangan dwi-aras** dilakukan dengan menggunakan *function* **ambang.m**

c. Kode Program **ambang.m**

```
function [G]=ambang(F,t)

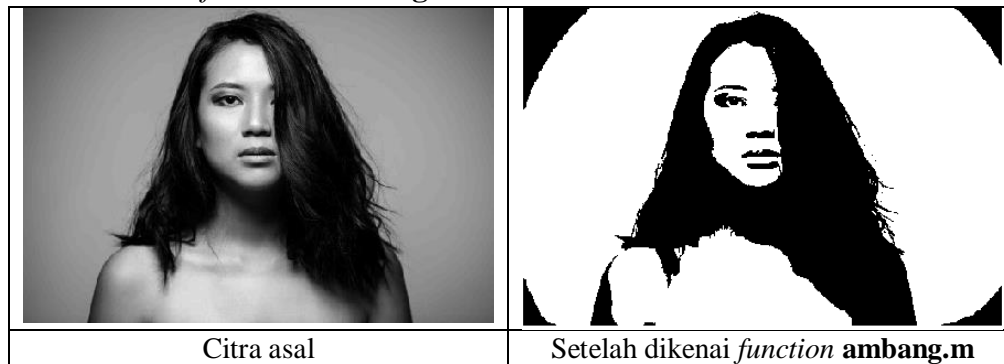
[m,n]=size(F);
for i=1:m
    for j=1:n
        if F(i,j)<=t
            G(i,j)=0;
        else
            G(i,j)=1;
        endif
    endfor
endfor
endfunction
```

**ambang.m**

d. Pemanggilan *function* **ambang.m**

```
>> K=ambang(abuabu,t);
```

e. Hasil *function* **ambang.m**



## 8. Melakukan pemotongan aras keabuan (potong.m)

Efek pemotongan (*clipping*) diperoleh dengan melakukan operasi

$$g(y, x) = \begin{cases} 0, & x \leq f_1 \\ f(y, x), & f_1 < f(y, x) < f_2 \\ 255, & x \geq f_2 \end{cases}$$

Nilai  $g$  di-nol-kan atau dipotong habis untuk intensitas asli dari 0 hingga  $f_1$  karena dipandang tidak mengandung informasi atau objek menarik. Demikian pula untuk nilai intensitas dari  $f_2$  ke atas yang mungkin hanya mengandung derau. **Pemotongan aras keabuan dapat dilakukan dengan *function* potong.m.**

a. Kode Program

```
function [Hasil] = potong(Img, f1, f2)

[tinggi, lebar] = size(Img);

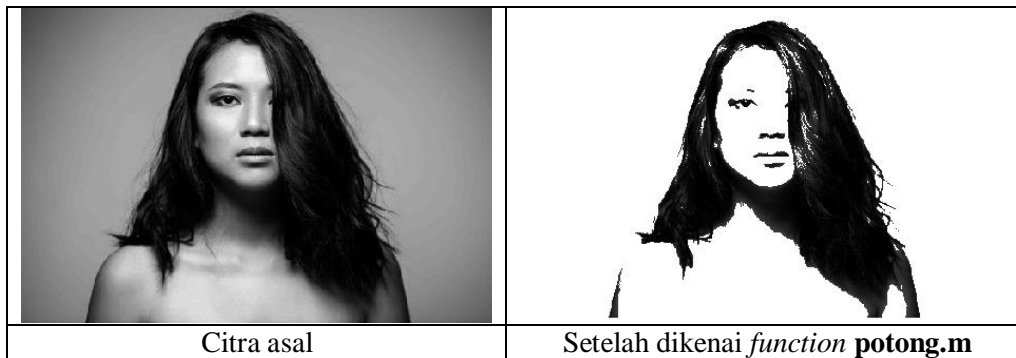
Hasil = Img;
for baris = 1:tinggi
    for kolom = 1:lebar
        if Hasil(baris, kolom) <= f1
            Hasil(baris, kolom) = 0;
        end
        if Hasil(baris, kolom) >= f2
            Hasil(baris, kolom) = 255;
        end
    end
end
end
```

potong.m

b. Pemanggilan *function* **potong.m**

```
>> L = potong(abuabu, 2, 62);
```

c. Hasil



## 9. Memberikan efek bergelombang pada citra (ripple.m)

Efek *Ripple* (riak) adalah aplikasi transformasi citra yang membuat gambar terlihat bergelombang. Efek riak dapat dibuat baik pada arah  $x$  maupun  $y$ . Transformasi efek riak adalah sebagai berikut:

$$x = x' + a_x \sin \frac{2\pi y'}{T_x}$$

$$y = y' + a_y \sin \frac{2\pi x'}{T_y}$$

Dalam hal ini,  $a_x$  dan  $a_y$  menyatakan amplitudo riak gelombang sinus, sedangkan  $T_x$  dan  $T_y$  menyatakan periode gelombang sinus. Efek *ripple* ini dapat dipanggil dengan *function* **ripple.m**.

a. Kode Program

```
function G=ripple(F,ax,ay,tx,ty)
dimensi=size(F);
tinggi=dimensi(1);
lebar=dimensi(2);
for y=1:tinggi
    for x=1:lebar
        x2=x+ax*sin(2*pi*y/tx);
        y2=y+ay*sin(2*pi*x/ty);
        if (x2>=1) && (x2<=lebar) && (y2>=1) && (y2<=tinggi)
            p=floor(y2);
            q=floor(x2);
            a=y2-p;
            b=x2-q;
            if (floor(x2)==lebar) || (floor(y2)==tinggi)
                G(y,x)=F(floor(y2),floor(x2));
            else
                intensitas=intensitas = (1-a)*((1-b)*F(p,q) + ...
                    b * F(p, q+1)) + ...
                    a * ((1-b) * F(p+1, q) + ...
                    b * F(p+1, q+1));
                G(y,x)=intensitas;
            endif
        else
            G(y,x)=0;
        endif
    endfor
endfor
G=uint8(G);
endfunction
```

ripple.m

b. Pemanggilan *function* **ripple.m**

```
>> M=ripple(abuabu,20,20,250,128);
```

c. Hasil



## 10. Membuat citra menjadi *blur* menggunakan *Butterworth Low Pass Filter* (blpf.m)

BLPF (*Butterworth Low Pass Filter*) merupakan jenis filter lolos-rendah yang digunakan untuk memperbaiki efek bergelombang yang dikenal dengan sebutan *ringing* yang diakibatkan oleh ILPF. Tidak seperti ILPF, BLPF ini tidak menghasilkan efek bergelombang ketika digunakan. BLPF dapat dipanggil dengan menggunakan *function* **blpf.m**.

a. Kode Program

```
function F=blpf(Img,d0,n)
Fs=double(Img);
[a,b]=size(Fs);
r=nextpow2(2*max(a,b));
p=2^r;
q=p;
u=0:(p-1);
v=0:(q-1);
idx=find(u>q/2);
u(idx)=u(idx)-q;
idy=find(v>p/2);
v(idy)=v(idy)-p;
[V,U]=meshgrid(v,u);
D=sqrt(V.^2+U.^2);
if nargin==2
    n=1;
endif
ambang=d0*p;
Hf=1./(1+D./ambang^(2*n));
Ff=fft2(Fs,p,q);
G=Hf.*Ff;
F=real(ifft2(G));
F=uint8(F(1:a,1:b));
endfunction
```

blpf.m

b. Pemanggilan *function* **blpf.m**

```
>> N=blpf(abuabu,0.5,0.2);
```

c. Hasil



## 11. Melakukan segmentasi warna pada citra (segwarna.m)

Segmentasi warna dapat dilakukan pada ruang warna HLS. Kemudian dilakukan pengubahan warna *Hue* yang berdekatan dengan warna yang menjadi pusat dalam fungsi keanggotaan *fuzzy*. Adapun nilai pada komponen *Luminance* dan *Saturation* disederhanakan menjadi tiga nilai yaitu 0, 128, dan 255. Warna yang mungkin timbul pada segmentasi warna ini ada 12 buah, yaitu:

Merah    Jingga    Kuning    Hijau    Cyan    Biru  
 Ungu    Magenta    Merah Muda    Hitam    **Putih**    Abu-abu

Setelah dilakukan pengubahan pada komponen *Hue*, *Luminance*, dan *Saturation*, warna HLS tadi diubah kembali menjadi RGB. Segmentasi warna pada citra dapat dilakukan dengan menggunakan *function segwarna.m*.

a. Kode Program

```

1 function [RGB]=segwarna(Img)
2 [tinggi,lebar,dim]=size(Img);
3 if dim<3
4     error('Masukan harus citra berwarna');
5 endif
6 [H,S,L]=RGBkeHSL(Img(:,:,1),Img(:,:,2),Img(:,:,3));
7 for y=1:tinggi
8     for x=1:lebar
9         h=H(y,x);
10        if h<11
11            h=0;
12        elseif h<32
13            h=21;
14        elseif h<54
15            h=43;
16        elseif h<116
17            h=85;
18        elseif h<141
19            h=128;
20        elseif h<185
21            h=170;
22        elseif h<202
23            h=191;
24        elseif h<223
25            h=213;
26        elseif h<244
27            h=234;
28        else
29            h=0;
30        endif
31        H(y,x)=h;
32        if S(y,x)>=200
33            S(y,x)=255;
22        elseif h<202
23            h=191;
24        elseif h<223
25            h=213;
26        elseif h<244
27            h=234;
28        else
29            h=0;
30        endif
31        H(y,x)=h;
32        if S(y,x)>=200
33            S(y,x)=255;
34        elseif S(y,x)<=20
35            S(y,x)=0;
36        else
37            S(y,x)=128;
38        endif
39        if L(y,x)>=200
40            L(y,x)=255;
41        elseif L(y,x)<=20
42            L(y,x)=0;
43        else
44            L(y,x)=128;
45        endif
46        endfor
47    endfor
48    [R,G,B]=HSLkeRGB(H,S,L);
49    RGB(:,:,1)=R;
50    RGB(:,:,2)=G;
51    RGB(:,:,3)=B;
52    return
53 endfunction
  
```

segwarna.m

b. Pemanggilan *function segwarna.m*

```
>> Z=segwarna(berwarna);
```

c. Hasil

