FINAL PROJECT
PENGOLAHAN CITRA DIGITAL

"MiniInstagram"

Dosen Pengampu:

Made Windu Antara Kesiman, S.T., M.Sc.

I Made Dendi Maysanjaya, S.Pd.,M.Eng

Oleh:

Dewa Ketut Satriawan Suditresna Jaya (1615051046 /5B)

Gede Angga Juliasta Wiguna (1615051072 /5B)

PENDIDIKAN TEKNIK INFORMATIKA

FAKULTAS TEKNIK DAN KEJURUAN

UNIVERSITAS PENDIDIKAN GANESHA

SINGARAJA

2018

InstaMini

5B_1615051046_1615051072

Pengolahan citra digital dalam aplikasinya dikehidupan nyata sangat banyak memiliki kebermanfaatan, mulai dari bidang pengolahan dokumen elektronik, bidang medis, kecerdasan buatan hingga bidang keamanan. Manfaat-manfaat tersebut tidak lain berasal dari dasar-dasar pengolahan citra yang kuat, untuk itu kami dalam final project mata kuliah **Pengolahan Citra Digital** membuat aplikasi *Mini Instagram* yang kami beri nama *InstaMini*. Dalam aplikasi sederhana yang kami buat ini sudah terdapat penerapan-penerapan dari proses pembelajaran yang kami terima selama proses perkuliahan berlangsung. InstaMini yang kami kembangkan memiliki 25 filter, kedua puluh lima filter tersebut berisi filter:

1. Citra Negatif
2. Citra GreyScale
3. Emboss
4. Freichen
5. Brightness
6. Roberts
7. Median
8. Contras
9. Laplacian
10. Kombinasi Flip
11. Sobel
12. Canny
13. Compass
14. Prewit
15. Threshold
16. Dilasi
17. Erosi
18. Opening
19. Closing
20. Bottom Hat
21. Top Hat
22. Filter Warna buatan sendiri Bernama LHSV
23. Filter Batik Buatan sendiri bernama My_Batik01
24. Filter Batik kedua Buatan sendiri bernama My_Batik02
25. Filter Batik ketiga Buatan sendiri bernama My_Batik03

Aplikasi InstaMini Ini selain memiliki 25 filter juga memiliki UI yang sudah kami kembangkan terhitung dari 3 kali pengembangan project, disamping itu juga kami meningkatkan performa aplikasi ketika saat mengganti filter, bernavigasi dan melakukan proses simpan serta pembatalan filter.

Release Note :

- V0.1.1 09-12-2018 - 6 Filter, File Image Must Same Folder
- V0.1.2 12-12-2018 - 18 Filter, Fix Bug Some Filter, Image Upload Anywhere can Up
- V0.1.3 27-12-2018 - 25 Filter, Fix Bug Very Heavy When Changes Filter, Increase UI Experiences

INSTRUKSI PENGGUNAAN APLIKASI:

1. Buka file gui.m yang tentunya dibuka menggunakan Matlab, Gunakan Versi terbaru jika ada.
2. Pada Menu Editor Tekan Tombol Run dengan icon panah Berwarna Hijau
3. Jika terdapat PopUp pesan Yang menyatakan bahwa gui.m tidak ditemukan pada current folder Sehingga anda perlu menekan tombol Change Folder untuk mengubah curent folder menjadi folder dimana file gui.m berada.
4. Tunggu Hingga Aplikasi GUI Mini Insta Muncul.
5. Ketika Aplikasi Sudah Berjalan, Lanjutkan dengan Memilih Photo/Citra Uji yang anda miliki (letak citra bisa bebas, Disarankan Menggunakan Citra Uji Lena dan Cam dengan Ekstensi .BMP)
6. Setelah Itu Tunggu Hingga Proses Pengolahan Citra dan pengolahan filter berjalan, Perlu ditekankan bahwa Lama proses pengolahan filter dipengaruhi oleh jenis dan kualitas citra uji, untuk citra uji lena.bmp yang merupakan citra true color RGB maka proses pengolahan citra ini akan memakan waktu yang cukup lama dibanding menggunakan cam.bmp. Namun ini hanya proses awal saja selanjutnya saat melakukan navigasi pemilihan filter serta menerapkan filter TIDAK AKAN MEMAKAN WAKTU. Ini merupakan kelebihan dari versi terbaru yang kami kembangkan.
7. Setelah Semua Filter Telah Muncul, dan proses pengolahan filter telah selesai selanjutnya anda bisa menerapkan 25 filter yang tersedia, mereset penerapan filter dengan cara menekan tombol refresh yang diwakili dengan icon panah memutar (Tombol Ini akan muncul hanya ketika anda menerapkan/ menekan salah satu tombol filter yang tersedia).
8. Jika Ingin Menyimpan hasil filter anda bisa menekan tombol Save yang berada di pojok kanan atas diwakili dengan Icon Panah Kebawah/icon Download (Tombol juga akan muncul ketika menekan salah satu dari beberapa tombol filter yang tersedia).

NOTE: Citra Uji Standar yang disarankan yaitu Citra Lena dan Cam dengan Ekstensi File .BMP

Berikut merupakan penjelasan serta code untuk masing-masing filter:

1. **CITRA NEGATIF**

   Penjelasan:
   Citra negatif merupakan citra yang nilai pikselnya berkebalikan dengan citra aslinya. Untuk citra grayscale 8-bit, apabila citra asli disimbolkan dengan I, maka negatif dari citra tersebut adalah I’ = 255-I.

   Code Function:

```
function img_out=filter_negatif(img_in)
[row, col, chan]=size(img_in);
img_out=uint8(zeros(size(img_in)));
if chan == 3
    r_chan=img_in(:,:,1);
    g_chan=img_in(:,:,2);
    b_chan=img_in(:,:,3);
```

```
    for i=1:row
        for j=1:col
            r_img(i,j)=255-r_chan(i,j);
            g_img(i,j)=255-g_chan(i,j);
            b_img(i,j)=255-b_chan(i,j);
        end
    end
    img_out(:,:,1)=r_img;
    img_out(:,:,2)=g_img;
    img_out(:,:,3)=b_img;
else
    for i=1:row
        for j=1:col
            img_out(i,j)=255-img_in(i,j);
        end
    end
end
filter_negatif=img_out;
%versi pendek
%img_out=255-img_in;
```

2. CITRA GRAYSCALE

Penjelasan:

Citra grayscale adalah citra yang nilai intensitas pikselnya berdasarkan derajat keabuan. Dalam filter ini akan mengkonversi citra RGB menjadi Greyscale dengan cara memecah setiap kanal warna RGB, baik itu Red Channel, Green Channel maupun Blue Channel. Kemudian masing-masing channel tiap piksel akan dikalikan dengan nilai standar derajat keabuan 0.299 untuk warna, 0.587 untuk merah hijau dan 0.114 untuk warna biru. Hasil dari perkalian tersebut kemudian dijumlahkan dan didapatkan citra grayscale.

Code Function:

```
Citra grayscale adalah citra yang nilai intensitas pikselnya
berdasarkan derajat keabuan.function img_out = filter_grey(image)
[row, col, chan]=size(image);
i = image;
if chan==3
        R = i(:, :, 1);
        G = i(:, :, 2);
        B = i(:, :, 3);
        img_out = zeros(row, col, 'uint8');

        for x=1:row
           for y=1:col
               img_out(x,y) = (R(x,y)*.3)+(G(x,y)*.6)+(B(x,y)*.1);
           end
        end
else
    img_out=i;
end
filter_grey = img_out;
```

3. **EMBOSS**

Penjelasan:

Embossing yaitu membuat citra seolah diukir pada permukaaan selembar nikel. Koefisien jendela konvolusi memiliki bobot tengah bernilai 0 & jumlah seluruh bobot = 0.

Code Function:

```
function img_in_out=filter_emboss(img_in)

mask = [0 0 -1 ; 0 0 0 ; 1 0 0];
[row, col] = size(mask);

[nrow, ncol, chan] = size(img_in);
img_in = cast(img_in, 'double');
newimg_in = zeros(nrow-1,ncol-1);

if chan==3
% Inner
    r_chan=img_in(:,:,1);
    g_chan=img_in(:,:,2);
    b_chan=img_in(:,:,3);
    r_chan=r_chan(1:nrow-1,1:ncol-1);
    g_chan=g_chan(1:nrow-1,1:ncol-1);
    b_chan=b_chan(1:nrow-1,1:ncol-1);
    for i = 0.5*(row+1) : nrow - 0.5*(row+1)
        for j = 0.5*(col+1) : ncol - 0.5*(col+1)
            subMat = img_in(i-1:i+1,j-1:j+1);
            r_chan(i,j) = sum(sum(subMat.*mask)) + 128;
            g_chan(i,j) = sum(sum(subMat.*mask)) + 128;
            b_chan(i,j) = sum(sum(subMat.*mask)) + 128;
        end
    end
    newimg_in(:,:,1)=r_chan;
    newimg_in(:,:,2)=g_chan;
    newimg_in(:,:,3)=b_chan;

else
    for i = 0.5*(row+1) : nrow - 0.5*(row+1)
        for j = 0.5*(col+1) : ncol - 0.5*(col+1)
            subMat = img_in(i-1:i+1,j-1:j+1);
            newimg_in(i,j) = sum(sum(subMat.*mask)) + 128;
        end
    end
end

newimg_in = cast(newimg_in, 'uint8');

img_in_out=newimg_in;
```

4. **FREICHEN**

Penjelasan:

Adalah salah satu operator yang digunakan untuk deteksi tepi pada citra, kadang disebut juga operator isotropik ditunjukkan di seperti pada gambar dibawah ini. Opetaror ini mirip seperti operator sobel, dengan dengan setiap angka 2 diganti menjadi akar 2.

Code Function:

```
function img_out=filter_freichen(img_in)
[row,col, chan]=size(img_in);
img_out=zeros(row-1,col-1);
akar2 = sqrt(2);

if chan==3
    img_in=filter_grey(img_in);
else

end

img_in = double(img_in);
    for i=1 : row-2
    for j=1 : col-2
        img_out(i, j) = sqrt(...
            (img_in(i,j+2)+akar2*img_in(i+1,j+2)+img_in(i+2,j+2) -
...
             img_in(i,j)-akar2*img_in(i+1,j)-img_in(i+2,j))^2 + ...
            (img_in(i,j)+akar2*img_in(i,j+1)+img_in(i,j+2) - ...
             img_in(i+2,j)-akar2*img_in(i+2,j+1)-img_in(i+2,j+2))^2);
    end
    end
    img_out=uint8(img_out);

filter_freichen=img_out;
```

5. BRIGHHNESS

Penjelasan:

Brightness adalah proses untuk kecerahan citra, jika intensitas pixel dikurangi dengan nilai tertentu maka citra akan menjadi lebih gelap, dan sebaliknya jika intensitas pixelnya ditambah dengan nilai tertentu maka akan lebih terang.

Code Function:

```
function img_out=filter_brightness(img_in,B)
%only for color_depth==8, citra grey
[row,col,chan]=size(img_in);
% height=size(img_in,1);
% width=size(img_in,2);
img_out=uint8(zeros(size(img_in)));
if chan ==3
    rColor = img_in(:, :, 1);
    gColor = img_in(:, :, 2);
    bColor = img_in(:, :, 3);
    for i = 1: row
        for j = 1: col
           rColor(i,j)=img_cliping(rColor(i,j)+B);
           gColor(i,j)=img_cliping(gColor(i,j)+B);
           bColor(i,j)=img_cliping(bColor(i,j)+B);
        end
    end
    img_out = cat(3, rColor, gColor, bColor);
else

    for i=1:row
        for j=1:col
            img_out(i,j)=img_cliping(img_in(i,j)+B);
```

```
        end
    end
end
filter_brightness=img_out;
```

6. **ROBERTS**

Penjelasan:

Operator Roberts merupakan suatu Teknik deteksi tepi sederhana dan memiliki tingkat komputasi yang cepat. Pada umumnya operator ini digunakan untuk citra grayscale. Operator Roberts dapat digambarkan dengan dua matriks berukuran 2 x 2 seperti berikut.

$$G_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Code Function:

```
function img_out=filter_roberts(img_in)
[row,col, chan]=size(img_in);
img_out=zeros(row-1,col-1);
if chan==3
    img_in=filter_grey(img_in);
end
img_in = double(img_in);
for i=1 : row-1
    for j=1 : col-1
        img_out(i,j)=sqrt((img_in(i,j)-
img_in(i+1,j+1))^2+(img_in(i+1,j)-img_in(i,j+1))^2);
    end
end

img_out=uint8(img_out);
filter_roberts=img_out;
```

7. **MEDIAN**

Penjelasan:

Filter median ini adalah teknik yang digunakan untuk mendapatkan citra baru dengan filter yang lebih baik daripada kontras dari citra asalnya.

Code Function:

```
function img_out=filter_median(img_in,window_size)
%window_size=ganjil;
[row, col, chan]=size(img_in);
border_size=(window_size-1)/2;
img_out=uint8(zeros(row-(2*border_size),col-(2*border_size)));
img_filter=uint8(zeros(window_size,window_size));
if chan==3
    r_chan=img_in(:,:,1);
    g_chan=img_in(:,:,2);
    b_chan=img_in(:,:,3);
    for i=border_size+1:row-border_size
    for j=border_size+1:col-border_size
        for i_img_filter=1:window_size
```

```
            for j_img_filter=1:window_size
                img_filter_r(i_img_filter,j_img_filter)=r_chan(i-
border_size+i_img_filter-1,j-border_size+j_img_filter-1);
                img_filter_g(i_img_filter,j_img_filter)=g_chan(i-
border_size+i_img_filter-1,j-border_size+j_img_filter-1);
                img_filter_b(i_img_filter,j_img_filter)=b_chan(i-
border_size+i_img_filter-1,j-border_size+j_img_filter-1);
            end
        end
        img_out_r(i-border_size,j-border_size)=mean(img_filter_r(:));
        img_out_g(i-border_size,j-border_size)=mean(img_filter_g(:));
        img_out_b(i-border_size,j-border_size)=mean(img_filter_b(:));
    end
    end
    img_out(:,:,1)=img_out_r;
    img_out(:,:,2)=img_out_g;
    img_out(:,:,3)=img_out_b;
else
    for i=border_size+1:row-border_size
    for j=border_size+1:col-border_size
        for i_img_filter=1:window_size
            for j_img_filter=1:window_size
                img_filter(i_img_filter,j_img_filter)=img_in(i-
border_size+i_img_filter-1,j-border_size+j_img_filter-1);
            end
        end
        img_out(i-border_size,j-border_size)=mean(img_filter(:));
    end
    end
end

filter_median=img_out;
```

8. CONTRAS

Penjelasan:

Kontras adalah tingkat penyebaran pixel – pixel ke dalan intensitas warna. Ada tiga macam kontras, yaitu kontras rendah, kontras tinggi, dan kontras normal. Dalam filter ini setiap kanal warna akan dikalikan dengan nilai skalar.

Code Function:

```
function img_out=filter_contras(img_in,G,P)
%only for color_depth==8, citra grey
[row,col,chan]=size(img_in);
% height=size(img_in,1);
% width=size(img_in,2);
img_out=uint8(zeros(size(img_in)));
if chan ==3
    rColor = img_in(:, :, 1);
    gColor = img_in(:, :, 2);
    bColor = img_in(:, :, 3);
    for i = 1: row
        for j = 1: col
           rColor(i,j)=img_cliping(G*(rColor(i,j)-P)+P);
           gColor(i,j)=img_cliping(G*(gColor(i,j)-P)+P);
           bColor(i,j)=img_cliping(G*(bColor(i,j)-P)+P);
        end
    end
    img_out = cat(3, rColor, gColor, bColor);
else
```

```
    for i=1:row
        for j=1:col
            img_out(i,j)=img_cliping(G*(img_in(i,j)-P)+P);
        end
    end
end
filter_contras=img_out;
```

9. LAPLACIAN

Penjelasan:

Filter ini merupakan filter dengan ukuran isotropic 2-D dari turunan kedua dari suatu citra. Penggunaan filter ini pada suatu citra akan menegaskan (highlights) wilayah-wilayah yang mengalami perubahan intensitas yang cepat dan oleh karena itu filter ini digunakan untuk deteksi tepi. Filterisasi Laplacian ini sering di kenakan kepada citra yang bertujuan untuk menghilangkan noise di citra tersebut.

Code Function:

```
function img_out = filter_laplacian(img_in)

[a, b, chan] = size(img_in);
img_out=zeros(a,b);
if chan==3
    img_in=filter_grey(img_in);
end
img_in=double(img_in);
for i=2 : a-1
    for j=2 : b-1
        img_out(i, j) = 8 * img_in(i,j)-(img_in(i-1,j-1) + img_in(i-
1,j)+ img_in(i-1,j+1) + img_in(i,j-1)+img_in(i,j+1)+img_in(i+1,j-1) +
img_in(i+1,j)+img_in(i+1,j+1)) ;

    end
end

img_out = uint8(img_out);
```

10. KOMBINASI FLIP

Penjelasan:

Flipping bertujuan untuk mendapatkan citra hasil pencerminan terhadap sumbu X (absis), sumbu Y (ordinat), maupun terhadap kedua sumbu tersebut sekaligus.

Code Function:

```
function img_out=flip_lena(img_in)
[row,col,chan]=size(img_in);
img_out=uint8(zeros(size(img_in))); %error when proceesing binary
image
% img_out=img_in;
center_row=row/2;
center_col=col/2;
img_resize=img_zoomout(img_in,2);
img_a=img_resize;
img_b=flip_image(img_resize,'H');
img_c=flip_image(img_resize, 'V');
img_d=flip_image(img_resize, 'HV');
if chan == 3
```

```
    r_chan_a=img_a(:,:,1);
    r_chan_b=img_b(:,:,1);
    r_chan_c=img_c(:,:,1);
    r_chan_d=img_d(:,:,1);

    g_chan_a=img_a(:,:,2);
    g_chan_b=img_b(:,:,2);
    g_chan_c=img_c(:,:,2);
    g_chan_d=img_d(:,:,2);

    b_chan_a=img_a(:,:,3);
    b_chan_b=img_b(:,:,3);
    b_chan_c=img_c(:,:,3);
    b_chan_d=img_d(:,:,3);

    for i=1 : row
        for j=1 : col
            if(j<=center_row && i<=center_col)
                out_r(i,j)=r_chan_a(i,j);
                out_g(i,j)=g_chan_a(i,j);
                out_b(i,j)=b_chan_a(i,j);
            elseif(j>center_row && i<=center_col)
                out_r(i,j)=r_chan_b(i,j-center_col);
                out_g(i,j)=g_chan_b(i,j-center_col);
                out_b(i,j)=b_chan_b(i,j-center_col);
            elseif(j<=center_row && i>center_col)
                out_r(i,j)=r_chan_c(i-center_row,j);
                out_g(i,j)=g_chan_c(i-center_row,j);
                out_b(i,j)=b_chan_c(i-center_row,j);
            elseif(j>center_row && i>center_col)
                out_r(i,j)=r_chan_d(i-center_row,j-center_col);
                out_g(i,j)=g_chan_d(i-center_row,j-center_col);
                out_b(i,j)=b_chan_d(i-center_row,j-center_col);
            end
        end
    end
    img_out(:,:,1)=out_r;
    img_out(:,:,2)=out_g;
    img_out(:,:,3)=out_b;
else
    for i=1 : row
        for j=1 : col
            if(j<=center_row && i<=center_col)
                img_out(i,j)=img_a(i,j);
            elseif(j>center_row && i<=center_col)
                img_out(i,j)=img_b(i,j-center_col);
            elseif(j<=center_row && i>center_col)
                img_out(i,j)=img_c(i-center_row,j);
            elseif(j>center_row && i>center_col)
                img_out(i,j)=img_d(i-center_row,j-center_col);
            end
        end
    end
end

flip_lena=img_out;
```

11. SOBEL

Penjelasan:

Sobel merupakan pengembangan metode robert dengan menggunakan filter HPF yang diberi satu angka nol penyangga. Metode ini mengambil prinsip dari fungsi laplacian dan gaussian yang dikenal sebagai fungsi untuk membangkitkan HPF. Kelebihan dari metode sobel ini adalah kemampuan untuk mengurangi noise sebelum melakukan perhitungan deteksi tepi.

Code Function:

```
function img_out=filter_sobel(img)
[a,b,chan]=size(img);
if chan==3
   img=rgbTogrey(img);
end

%penentuan nilai Kernel
sobelhor = [-1 0 1; -2 0 2; -1 0 1];
sobelver = [-1 -2 -1; 0 0 0; 1 2 1];

%deteksi kordinat x
for i=2:a-1
    for j=2: b-1
        y=i-1;
        x=j-1;
        pim=double(img(y:y+2,x:x+2));
        Horizontal(i,j)=sum(sum(pim.*sobelhor));
        Vertikal(i,j)=sum(sum(pim.*sobelver));
    end
end
img_out=uint8(sqrt((Horizontal.^2)+(Vertikal.^2)));
filter_sobel=img_out;
```

12. CANNY

Penjelasan:

John F. Canny pada tahun 1986 mengembangkan metode canny dengan menggunakanalgoritma multi-tahap untuk mendeteksi berbagai tepi dalam gambar. Kategori algoritma yang dikembangkan adalah:

a. Deteksi: Kemungkinan mendeteksi tepi yang benar harus dimaksimalkan sedangkan kemungkinan mendeteksi tepi yang salah harus diminimalkan. Hal ini bertujuan untuk memaksimalkan rasio signal to noise
b. Lokalisasi: tepi yang terdeteksi harus sedekat mungkin dengan tepi nyata.
c. Jumlah tanggapan: satu tepi nyata tidak harus menghasilkan lebih dari satu ujung yang terdeteksi

Code Function:

```
function img_out = filter_canny(F, ambang_bawah, ambang_atas)
[row,col,chan]=size(F);
if chan==3
    F=rgbTogrey(F);
end
% Pemerolehan tepi objek pada citra F
%    melalui operator Canny
% Argumen:
```

```
%    ambang_bawah = batas bawah untuk ambang histeresis
%                   Nilai bawaan 011
%    ambang_atas = batas atas untuk ambang histeresis
%                   Nilai bawaan 0,3
% Hasil: citra G

% Menentukan nilai ambang bawaan
if nargin < 2
    ambang_bawah = 0.1;
end

if nargin < 2
    ambang_atas = 0.3;
end

% Kernel Gaussians
HG = [1  4  6  4  1
      4 16 24 16  4
      6 24 36 24  6
      4 16 24 16  4
      1  4  6  4  1 ] / 256.0;
[hHG, wHG] = size(HG);
h2 = floor(hHG / 2);
w2 = floor(wHG / 2);

% Kenakan operasi Gaussian
G = double(img_deteksi(F, HG, true));

% Pastikan hasilnya berada antara 0 sampai dengan 255
[m, n] = size(G);
for i = 1 : m
    for j = 1 : n
        G(i, j) = round(G(i, j));

        if G(i, j) > 255
            G(i, j) = 255;
        else
            if G(i, j) < 0
                G(i, j) = 0;
            end
        end
    end
end

% Kenakan perhitungan gradien dan arah tepi
Theta = zeros(m, n);
Grad = zeros(m, n);
for i = 1 : m-1
    for j = 1 : n-1
        gx = (G(i,j+1)-G(i,j) + ...
              G(i+1,j+1)-G(i+1,j)) / 2;
        gy = (G(i,j)-G(i+1,j) + ...
              G(i,j+1)-G(i+1,j+1)) / 2;
        Grad(i, j) = sqrt(gx.^2 + gy.^2);
        Theta(i,j) = atan2(gy, gx);
    end
end

% Konversi arah tepi menjadi 0, 45, 90, atau 135 derajat
[r c] = size (Theta);
```

```
if Theta < 0
    Theta = Theta + pi; % Jangkauan menjadi 0 s/d pi
end

for i = 1 : r
    for j = 1 : c
        if (Theta(i,j) < pi/8 || Theta(i,j) >= 7/8*pi)
            Theta(i,j) = 0;
        elseif (Theta(i,j)>=pi/8 && Theta(i,j) < 3*pi/8 )
            Theta(i,j) = 45;
        elseif (Theta(i,j) >=3*pi/8 && Theta(i,j) < 5*pi/8 )
            Theta(i,j) = 90;
        else
            Theta(i,j) = 135;
        end
    end
end

% penghilangan non-maksimum
Non_max = Grad;

for i = 1+h2 : r-h2
    for j = 1+w2 : c-h2
        if Theta(i,j) == 0
            if (Grad(i,j) <= Grad(i,j+1)) || ...
               (Grad(i,j)<= Grad(i,j-1))
                Non_max(i,j) = 0;
            end
        elseif Theta(i,j) == 45
            if (Grad(i,j) <= Grad(i-1,j+1)) || ...
               (Grad(i,j) <= Grad(i+1,j-1))
                Non_max(i,j) = 0;
            end
        elseif Theta(i,j) == 90
            if (Grad(i,j) <= Grad(i+1,j) ) || ...
               (Grad(i,j) <= Grad(i-1,j))
                Non_max(i,j) = 0;
            end
        else
            if (Grad(i,j) <= Grad(i+1,j+1)) || ...
               (Grad(i,j) <= Grad(i-1,j-1))
                Non_max(i,j) = 0;
            end
        end
    end
end

% Pengambangan histeresis
ambang_bawah = ambang_bawah * max(max(Non_max));
ambang_atas =  ambang_atas * max(max(Non_max));

Histeresis = Non_max;

% ----- Penentuan awal untuk memberikan nilai
% ----- 0, 128, dan 255
for i = 1+h2 : r-h2
    for j = 1+w2 : c-w2
        if (Histeresis(i,j) >= ambang_atas)
            Histeresis(i,j) = 255;
        end
```

```
        if (Histeresis(i,j) < ambang_atas) && ...
           (Histeresis(i,j) >= ambang_bawah)
            Histeresis(i,j)= 128;
        end

        if (Histeresis(i,j) < ambang_bawah)
            Histeresis(i,j) = 0;
        end
    end
end

% ----- Penggantian angka 128 menjadi 255
% ----- Berakhir kalau tidak ada lagi yang berubah
ulang = true;
while ulang
    ulang = false;
    for i = 1+h2 : r-h2
        for j = 1+w2 : c-w2
            if (Histeresis(i,j) == 128)
                if (Histeresis(i-1, j-1) == 255) && ...
                   (Histeresis(i-1, j) == 255) && ...
                   (Histeresis(i, j+1) == 255) && ...
                   (Histeresis(i, j-1) == 255) && ...
                   (Histeresis(i, j+1) == 255) && ...
                   (Histeresis(i+1, j-1) == 255) && ...
                   (Histeresis(i+1, j) == 255) && ...
                   (Histeresis(i+1, j+1) == 255)
                    Histeresis(i,j) = 255;

                    ulang = true;
                end
            end
        end
    end
end

% ----- Penggantian angka 128 menjadi 0
% ----- untuk yang tersisa
for i = 1+h2 : r-h2
    for j = 1+w2 : c-w2
        if (Histeresis(i,j) == 128)
            Histeresis(i,j) = 0;
        end
    end
end

% Buang tepi
for i = 1+h2 : r-h2
    for j = 1+w2 : c-w2
        img_out(i-1,j-1) = Histeresis(i,j);
    end
end

filter_canny = img_out;
```

13. COMPASS

Penjelasan:

Fiter ini digunakan untuk mendeteksi semutepi dari berbagai arah di dalam citra. Operator kompas yang digunakan menampilkan tepi dari 8 macam arah mata angin

Code Function:

```
function img_out=filter_compass(img)
[a,b,chan]=size(img);
if chan==3
   img=rgbTogrey(img);
end
%penentuan nilai Kernel
utara = [1 1 1; 1 -2 1; -1 -1 -1];
selatan = [-1 -1 -1; 1 -2 1; 1 1 1];
timur = [-1 1 1; -1 -2 1; -1 1 1];
barat = [1 1 -1; 1 -2 -1; 1 1 -1];

for i=2:a-1
    for j=2: b-1
        y=i-1;
        x=j-1;
        pim=double(img(y:y+2,x:x+2));
        Barat(i,j)=sum(sum(pim.*barat));
        Utara(i,j)=sum(sum(pim.*utara));
        Timur(i,j)=sum(sum(pim.*timur));
        Selatan(i,j)=sum(sum(pim.*selatan));
    end
end
img_out=uint8(sqrt((Utara.^2)+(Timur.^2)+(Barat.^2)+(Selatan.^2)));
filter_compass = img_out;
```

14. PREWIT

Penjelasan:

Prewitt adalah pengembangan dari metode robert dengan menggunakan filter HPF yang diberi satu angka nol penyangga. Metode prewitt mengambil prinsip dari fungsi laplacian yang dikenal sebagai fungsi untuk membangkitkan HPF.

Code Function:

```
function img_out=filter_prewit(img)
[a,b,chan]=size(img);
if chan==3
   img=rgbTogrey(img);
end
%penentuan nilai Kernel
prewitthor = [-1 0 1; -1 0 1; -1 0 1];
prewittver = [-1 -1 -1; 0 0 0; 1 1 1];

for i=2:a-1
    for j=2: b-1
        y=i-1;
        x=j-1;
        pim=double(img(y:y+2,x:x+2));
        Horizontal(i,j)=sum(sum(pim.*prewitthor));
        Vertikal(i,j)=sum(sum(pim.*prewittver));
    end
end
img_out=uint8(sqrt((Horizontal.^2)+(Vertikal.^2)));
filter_prewit=img_out;
```

15. THRESHOLD

Penjelasan:

Pada operasi thresholding, level intensitas sebuah piksel akan dipetakan ke salah satu dari dua buah nilai a1 atau a2, berdasarkan sebuah nilai ambang (threshold) T

Code Function:

```
function img_out = filter_threshold(img_in, T)
[row, col, chan] =size(img_in);
    %perulangan untuk mencari nilai pixel tiap baris dan kolom citra
    if chan==3
       img_in=rgbTogrey(img_in);
    end
    for i=1 :row
        for j=1 :col
            %jika nilai pixel dari citra lebih besar sama dengan
nilai T
            %maka tentukan nilai array a menjadi 255;
            if (img_in(i,j) >= T)
                img_out(i,j)=255;
            %jika nilai pixel dari citra lebih kecil nilai T
            %maka tentukan nilai array a menjadi 0;
            else
                img_out(i,j)=0;
            end
        end
    end
filter_threshold = img_out;
```

16. DILASI

Penjelasan:

Filter yang ke 16 ini merupakan filter dimana ia berfungsi untuk memperbesar segmen objek (citra biner) dengan menambah lapisan disekeliling objek. Atau dengan menjadi titik latar (0) yang bertetangga dengan titik objek (1) berubah menjadi titik objek (1).

Code function:

```
function MOutput=filter_dilasi(MInput,window_size)
[row,col,chan]=size(MInput);
%window_size=ganjil;
border_size=(window_size-1)/2;
%SetLength(MOutput,Length(MInput)-(2*border_size),Length(MInput[0])-
(2*border_size));
MOutput=uint8(zeros(size(MInput,1)-(2*border_size),size(MInput,2)-
(2*border_size)));
%SetLength(MFilter,window_size,windows_size);
MFilter=uint8(zeros(window_size,window_size));

if chan==3
  MInput=rgbTogrey(MInput);
end
    for i=border_size+1:size(MInput,1)-border_size
        for j=border_size+1:size(MInput,2)-border_size
            for i_mfilter=1:window_size
                for j_mfilter=1:window_size
                    MFilter(i_mfilter,j_mfilter)=MInput(i-
border_size+i_mfilter-1,j-border_size+j_mfilter-1);
                end
```

```
            end
            MOutput(i-border_size,j-border_size)=min(MFilter(:));
        end
    end

filter_dilasi=MOutput;
```

17. **EROSI**

Penjelasan:

Erosi atau pengikisan adalah kebalikan dari dilasi yaitu teknik yang bertujuan untuk memperkecil atau mengikis tepi objek. Atau dengan menjadi titik objek (1) yang bertetangga dengan titik latar (0) menjadi titik latar (0).

Code Function:

```
function MOutput=filter_erosi(MInput,window_size)
[row,col,chan]=size(MInput);
%window_size=ganjil;
border_size=(window_size-1)/2;
MOutput=uint8(zeros(size(MInput,1)-(2*border_size),size(MInput,2)-
(2*border_size)));
MFilter=uint8(zeros(window_size,window_size));

if chan==3
    MInput=rgbTogrey(MInput);
end
    for i=border_size+1:size(MInput,1)-border_size
        for j=border_size+1:size(MInput,2)-border_size
            for i_mfilter=1:window_size
                for j_mfilter=1:window_size
                    MFilter(i_mfilter,j_mfilter)=MInput(i-
border_size+i_mfilter-1,j-border_size+j_mfilter-1);
                end
            end
            MOutput(i-border_size,j-border_size)=max(MFilter(:));
        end
    end

filter_erosi=MOutput;
```

18. **OPENING**

Penjelasan:

Opening ialah proses erosi yangmana juga diikuti dengan proses dilasi.Dimulai dengan melakukan proses erosi pada citra yang kemudian hasil dari proses tersebut kembali dilakukan proses dilasi. Proses opening biasanya digunakan untuk menghilangkan objek-objek kecil yang kurus serta dapat membuat tepi citra lebih smooth (untuk citra berukuran besar).

Code Function:

```
function img_out=bukaTutup(img_in, jum,pilihan)
if pilihan==1
    %opening
    img_out=filter_dilasi((filter_erosi(img_in, jum)),jum);
elseif pilihan==0
    %closing
```

```
    img_out=filter_erosi((filter_dilasi(img_in, jum)),jum);
end

bukaTutup=img_out;
```

19. CLOSING

Penjelasan:

Closing merupakan kebalikan dari proses opening. Dimana cara kerja filter ini yaitu citra terlebih dahulu dilakukan dilasi yang kemudian dilanjutkan dengan proses erosi pada citra hasil dilasi. Proses Closing bertujuan untuk mengisi lubang kecil pada objek, serta untuk menggabungkan objek yang berdekatan

Code Function:

```
function img_out=bukaTutup(img_in, jum,pilihan)
if pilihan==1
    %opening
    img_out=filter_dilasi((filter_erosi(img_in, jum)),jum);
elseif pilihan==0
    %closing
    img_out=filter_erosi((filter_dilasi(img_in, jum)),jum);
end

bukaTutup=img_out;
```

20. BOTTOM HAT

Penjelasan:

Secara prinsip, operasi bottom hat ini memperbesar warna putih melalui dilasi, diikuti dengan pengecilan warna putih melalui erosi dan kemudian dikurangi dengan citra asal atau citra masukan. Dilasi yang diikuti dengan erosi memberikan dampak berupa objek-objek yang berdekatan menjadi semakin dekat. Sementara pengurangan oleh citra asal membuat penghubung antar objek menjadi hasil yang tersisa. Dengan demikian, hasil yang tersisa adalah piksel-piksel yang digunakan untuk mengisi “lubang”, atau “penghubung objek”.

Code Function:

```
function img_out=filter_bottomhat(img_in,ukuran)
[row,col,chan]=size(img_in);
if chan==3
    img_in=rgbTogrey(img_in);
end
clossing=bukaTutup(img_in,ukuran,0);
row=size(clossing,1);
col=size(clossing,2);
img_out = imresize(clossing, [col,row]) - imresize(img_in,
[col,row]);

filter_bottomhat=img_out;
```

21. TOP HAT

Penjelasan:

Transformasi Top-Hat didefinisikan sebagai perbedaan antara citra dan citra setelah mengalami operasi opening . Transformasi ini berguna untuk mendapatkan bentuk global suatu objek yang mempunyai intensitas yang bervariasi.

Code Function:

```
function img_out=filter_tophat(img_in,ukuran)
[row,col,chan]=size(img_in);
if chan==3
    img_in=rgbTogrey(img_in);
end
opening=bukaTutup(img_in,ukuran,1);
row=size(opening,1);
col=size(opening,2);

img_out = imresize(img_in, [col,row]) - imresize(opening, [col,row]);

filter_tophat=img_out;
```

22. FILTER WARNA BUATAN SENDIRI LHSV

Penjelasan:

Filter ke 22 ini merupakan filter yang menerapkan function konversi dari rgb ke HSV selanjutnya setelah mendapatkan nilai Hue, Saturnation dan Value ketiga data ini akan diolah dimana data Huenya setiap piksel akan dikalikan dengan 3, saturnation setiap piksel dikalikan dengan nilai 4 sedangkan value setiap pikselnya akan dikalikan dengan 0.5. Selanjutnya adalah tahap konversi ke bentuk RGB.

Code Function:

```
function img_out=filter_Lhsv(img_in)
[row,col,chan]=size(img_in);
if chan==3
   [H,S,V]=rgbToHSV(img_in);
   hue=H;
   saturnation=S;
   value=V;
   for i=1:row
       for j=1:col
           hue(i,j)=hue(i,j)*3;
           saturnation(i,j)=saturnation(i,j)*4;
           value(i,j)=value(i,j)*0.5;
       end
   end
   H=hue;
   S=saturnation;
   [R,G,B]=hsvToRGB(H,S,V);
   img_out=cat(3,R,G,B);
   img_out=uint8(img_out);
else
    img_out=img_in;
end
```

```
filter_Lhsv=img_out;
```

23. FILTER BATIK 01

Penjelasan:

Filter ini merupakan filter buatan sendiri dimana filter ini juga merupakan komninasi dari berbagai function yang telah dibuat seperti function flip, rotasi zoomout dan lain sebagainya. Mula-mula image masukan akan dilakukan flip menjadi 4 jenis image. Pertama adalah image original tanpa flip kedua adalah flip vertikal, ketika flip Horizontal dan keempat adalah flip horizontal vertikal. Keempat image selanjutnya di gabungkan sehingga menjadi image cermin, image ini lah menjadi dasar pengolahan image selanjutnya dimana image ini akan dioleh dengan function img_posisi function ini telah kami buat khusus untuk menghandle peletakan image yang juga dilakukan pengecilan image, dengan function ini kita dapat mudah meletakkan gambar baru diatas gambar dasar yang mana posisinya dapat kita atur, apakah berada di pojok kiri atas/bawah pojok kanan atas/bawah bahkan ditengah. Dengan function inilah kami kombinasikan dengan image masukan dan function lainnya sehingga membentuk pola batik sederhana.

Code Function:

```
function img_out=filter_batik1(img_in)
dasar=uint8(zeros(size(img_in)));

dasar1=flip_lena(img_in);
dasar2=img_posisi(dasar,dasar1,2,1);
dasar3=img_posisi(dasar2,dasar1,2,2);
dasar4=img_posisi(dasar3,dasar1,2,3);
dasar5=img_posisi(dasar4,dasar1,2,4);
dasar6=img_posisi(dasar5,dasar1,2,5);

dasar7=img_posisi(dasar,dasar6,2,1);
dasar8=img_posisi(dasar7,dasar6,2,2);
dasar9=img_posisi(dasar8,dasar6,2,3);
dasar10=img_posisi(dasar9,dasar6,2,4);

dasar11=img_posisi(dasar,dasar10,2,1);
dasar12=img_posisi(dasar11,dasar10,2,2);
dasar13=img_posisi(dasar12,dasar10,2,3);
dasar14=img_posisi(dasar13,dasar10,2,4);

b=len_d(img_in);
a=flip_image(b,'H');
d=len_a(img_in);
c=flip_image(d,'H');
tengah=flip_lena(img_in);
tengah2=rotasi_cw(tengah,1);
tengah3=img_posisi(tengah2,tengah,2,5);
tengah=img_posisi(tengah,tengah3,2,5);

d1=img_posisi(dasar14,a,2,1);
d2=img_posisi(d1,b,2,2);
d3=img_posisi(d2,c,2,3);
d4=img_posisi(d3,d,2,4);
d5=img_posisi(d4,tengah,2,5);
```

```
h=img_posisi(dasar,d5,2,1);
h=img_posisi(h,d5,2,2);
h=img_posisi(h,d5,2,3);
img_out=img_posisi(h,d5,2,4);

filter_batik1=img_out;
```

24. FILTER BATIK 02

Penjelasan:

Sama Seperti filter batik 01 filter batik kedua ini juga mengkombinasikan beberapa function buatan kami sendiri bedanya adalah cita hasil pencerminan sebelum digunakan untuk diproses menggunakan function img_posisi untuk membuat pola dengan meletakkan posisi image tersebet, terlebih dahulu di filter dengan menggunakan filter citra negatif sehingga nampak hasil dari citra ini berupa pola dengan warna biru sebagai efek dari filter citra negatif (jika citra masukannya adalah lena.bmp). Satu lagi dibagian function img_posisi parameter masukan tersebut memiliki penjelasan yaitu img_posisi(image masukan yang akan menjadi dasar/canvas, image yang akan mengalami pengecilan dan diletakan di atas image dasar yang posisinya bisa diatur, masukan angka untuk dijadikan pengecilan image jika image sebelumya sudah dikecilkan dan dalam function ini tidak ingin dikecilkan lagi cukup input angka 0 untuk tidak memperkecil gambar, parameter terakhir adalah indikator lrtak image yang akan diletakan diatas image dasar 1 untuk pojok kiri atas, 2 pojok kanan atas, 3 pojok kanan bawah , 4 pojok kiri bawah dan 5 untuk posisi tengah.

Code Function:

```
function img_out=filter_batik2(img_in)
dasar=uint8(zeros(size(img_in)));
dasar1=flip_lena(img_in);
dasar1=filter_negatif(dasar1);
dasar2=img_posisi(dasar,dasar1,2,1);
dasar3=img_posisi(dasar2,dasar1,2,2);
dasar4=img_posisi(dasar3,dasar1,2,3);
dasar5=img_posisi(dasar4,dasar1,2,4);
dasar6=img_posisi(dasar5,dasar1,2,5);

dasar7=img_posisi(dasar,dasar6,2,1);
dasar8=img_posisi(dasar7,dasar6,2,2);
dasar9=img_posisi(dasar8,dasar6,2,3);
dasar10=img_posisi(dasar9,dasar6,2,4);

dasar11=img_posisi(dasar,dasar10,2,1);
dasar12=img_posisi(dasar11,dasar10,2,2);
dasar13=img_posisi(dasar12,dasar10,2,3);
dasar14=img_posisi(dasar13,dasar10,2,4);
img_out=dasar14;
filter_batik2=img_out;
```

25. FILTER BATIK 03

Penjelasan:

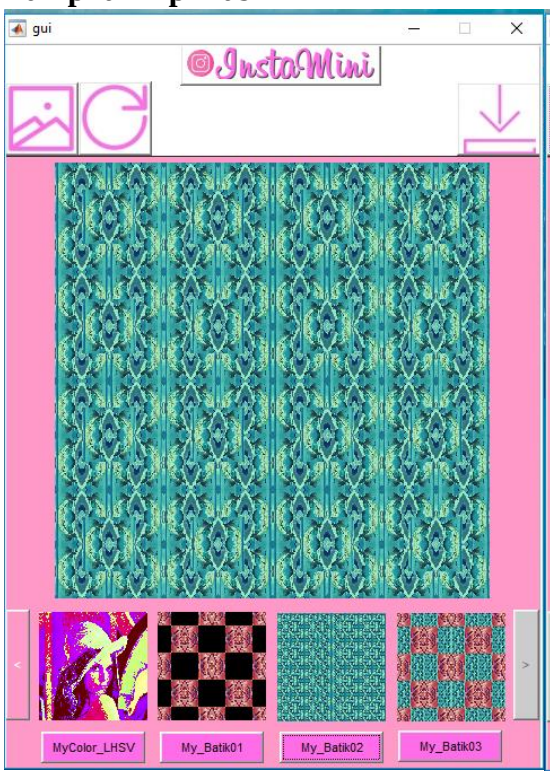
Filter terakhir ini merupakan filter gabungan atau kombinasi dari filter Batik 01 dengan filter batik 02. Sehingga dari kedua filter tersebut didapat image dengan pola unik (jika lena.bmp akan tidak ada warna hitam sebagai latar lagi namun sudah digantikan dengan image batik 02).

Code Function:

```
function img_out=filter_batik11(img_in)
dasar=uint8(zeros(size(img_in)));

dasar1=flip_lena(img_in);
dasar1=filter_negatif(dasar1);
dasar2=img_posisi(dasar,dasar1,2,1);
dasar3=img_posisi(dasar2,dasar1,2,2);
dasar4=img_posisi(dasar3,dasar1,2,3);
dasar5=img_posisi(dasar4,dasar1,2,4);
dasar6=img_posisi(dasar5,dasar1,2,5);

dasar7=img_posisi(dasar,dasar6,2,1);
dasar8=img_posisi(dasar7,dasar6,2,2);
dasar9=img_posisi(dasar8,dasar6,2,3);
dasar10=img_posisi(dasar9,dasar6,2,4);

dasar11=img_posisi(dasar,dasar10,2,1);
dasar12=img_posisi(dasar11,dasar10,2,2);
dasar13=img_posisi(dasar12,dasar10,2,3);
dasar14=img_posisi(dasar13,dasar10,2,4);

b=len_d_neg(img_in);
a=flip_image(b,'H');
d=len_a_neg(img_in);
c=flip_image(d,'H');
tengah=flip_lena(img_in);
tengah2=rotasi_cw(tengah,1);
tengah3=img_posisi(tengah2,tengah,2,5);
tengah=img_posisi(tengah,tengah3,2,5);

d1=img_posisi(dasar14,a,2,1);
d2=img_posisi(d1,b,2,2);
d3=img_posisi(d2,c,2,3);
d4=img_posisi(d3,d,2,4);
d5=img_posisi(d4,tengah,2,5);
h=img_posisi(dasar,d5,2,1);
h=img_posisi(h,d5,2,2);
h=img_posisi(h,d5,2,3);
img_out=img_posisi(h,d5,2,4);

filter_batik11=img_out;
```

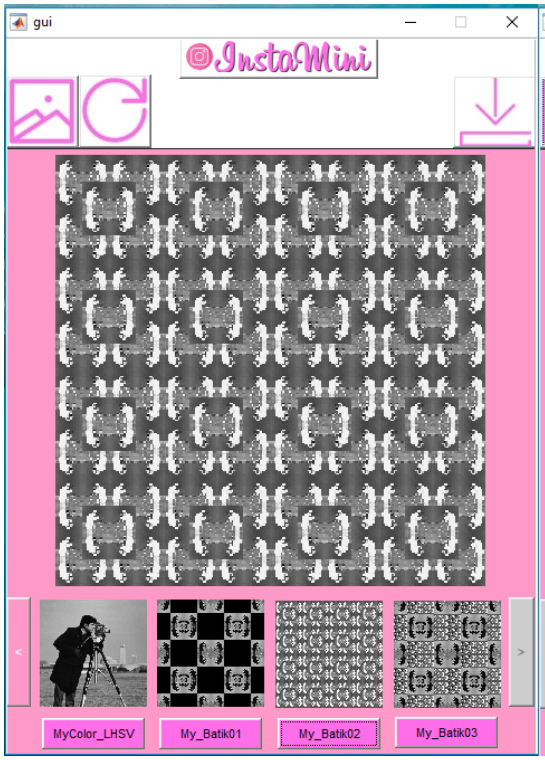
LAMPIRAN-LAMPIRAN

1. Tampilan Aplikasi

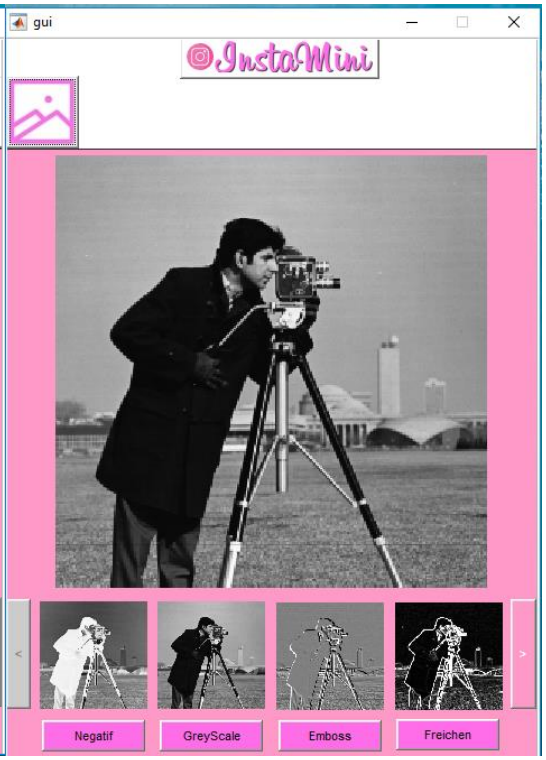


Tampilan Image Lena saat Di terapkan salah satu Filter

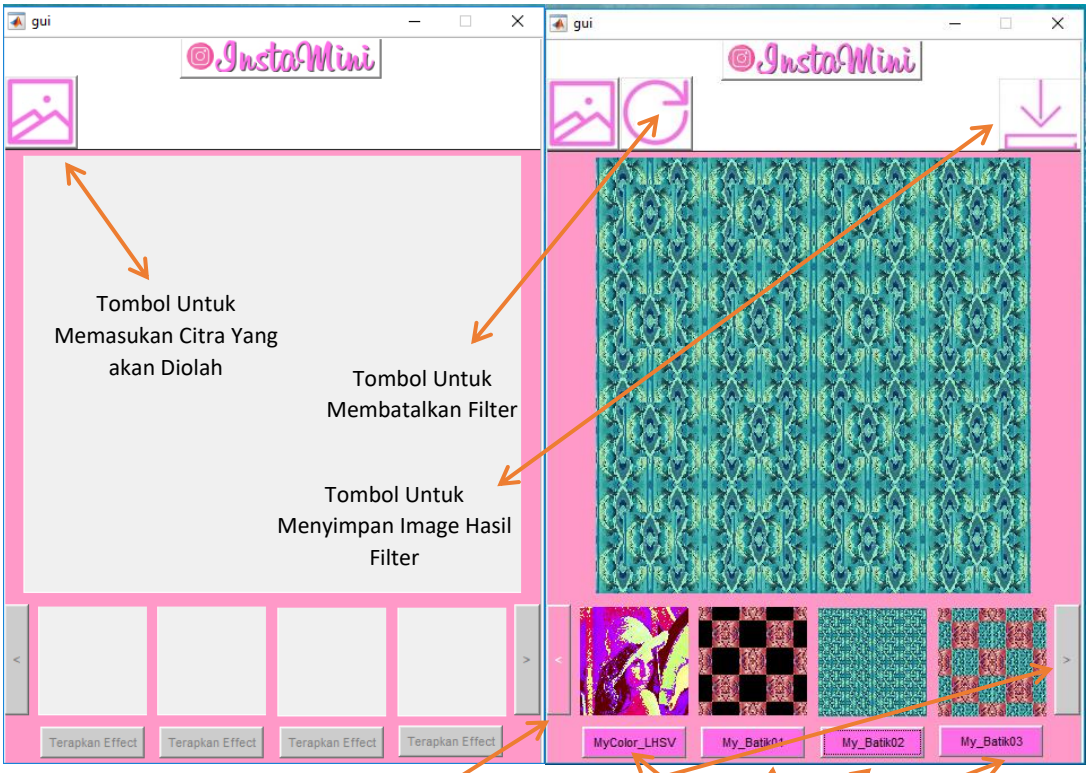
Tampilan Awal Aplikasi Saat meload Gambar True Color Citra Lena



Tampilan Image Cam saat Di terapkan salah satu Filter



Tampilan Awal Aplikasi Saat meload Gambar Grayscale Citra Cam



Tampilan Interface Awal Aplikasi Sebelum Meload Gambar

Tampilan Interface Setelah menerapkan Filter