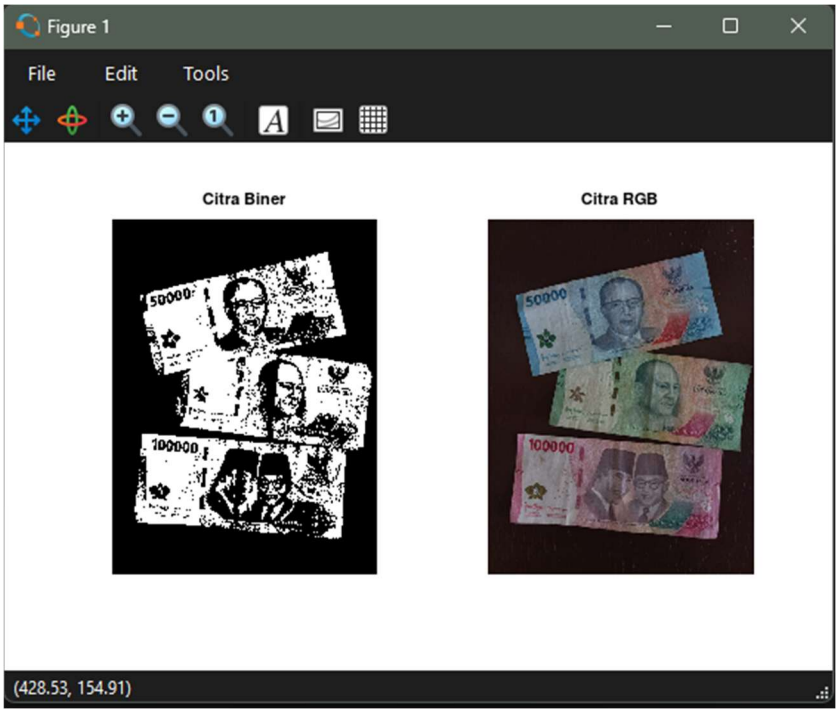
UTS Pengolahan Citra Semester 4 2025

Nama : Muhammad Rizal Indriawan

Nim : 23552010-19

1. Jenis citra
   a. Citra Biner

   Citra biner adalah citra dengan setiap piksel hanya dinyatakan dengan sebuah nilai dari dua buah kemungkinan (yaitu nilai 0 dan 1).



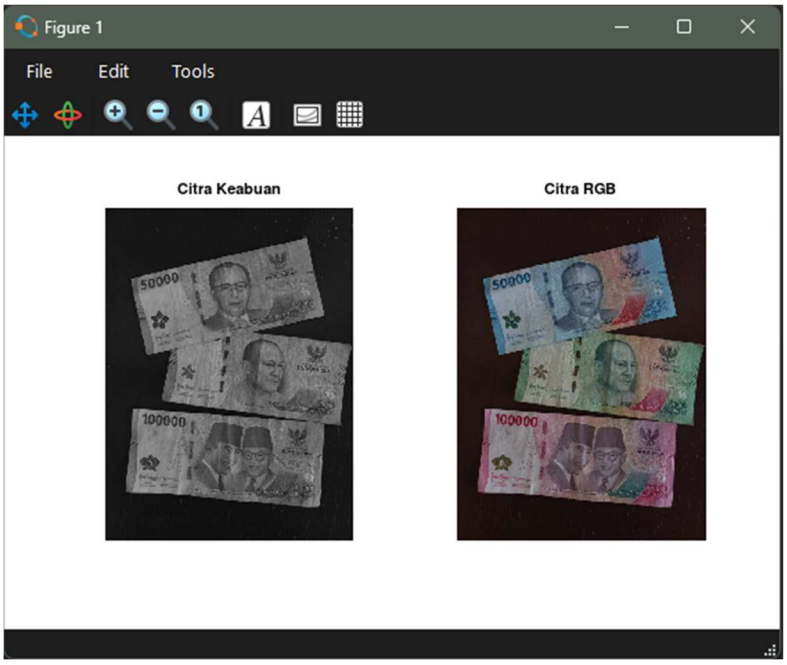
Code :

```
pkg load image;

img = imread('E:\coolyeah\pengolahan citra\gambar\citraRGB.jpeg');
% Ubah ke grayscale terlebih dahulu
gray_img = rgb2gray(img);
[tinggi, lebar] = size(gray_img);
ambang = 100; % threshold
biner = zeros(tinggi, lebar);
for baris = 1:tinggi
for kolom = 1:lebar
if gray_img(baris, kolom) >= ambang
biner(baris, kolom) = 1;
else
biner(baris, kolom) = 0;
end
end
end
```

```
subplot (1,2,1); imshow(biner); title ('Citra Biner');
subplot (1,2,2); imshow (img); title ('Citra RGB');
```

b. Citra Grayscale

citra jenis ini menangani gradasi warna hitam dan putih, yang tentu saja menghasilkan efek warna abu-abu. Pada jenis gambar ini, warna dinyatakan dengan intensitas.

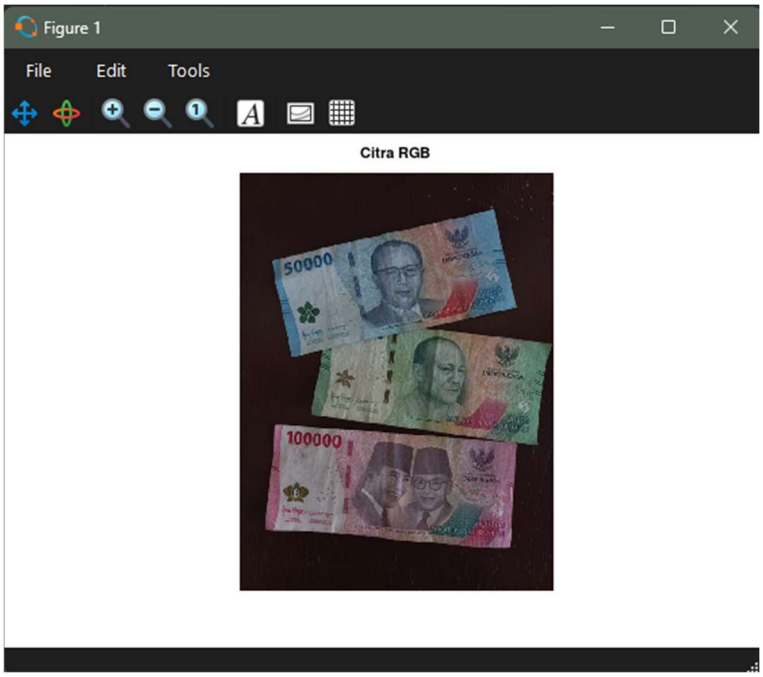


Code :

```
img = imread('E:\coolyeah\pengolahan citra\gambar\citraRGB.jpeg');
abu = rgb2gray(img);
subplot (1,2,1); imshow(abu); title ('Citra Keabuan');
subplot (1,2,2); imshow (img); title ('Citra RGB');
```

c. Citra RGB

citra RGB, merupakan jenis citra yang menyajikan warna dalam bentuk komponen R (merah), G (hijau), dan B (biru).



Code :

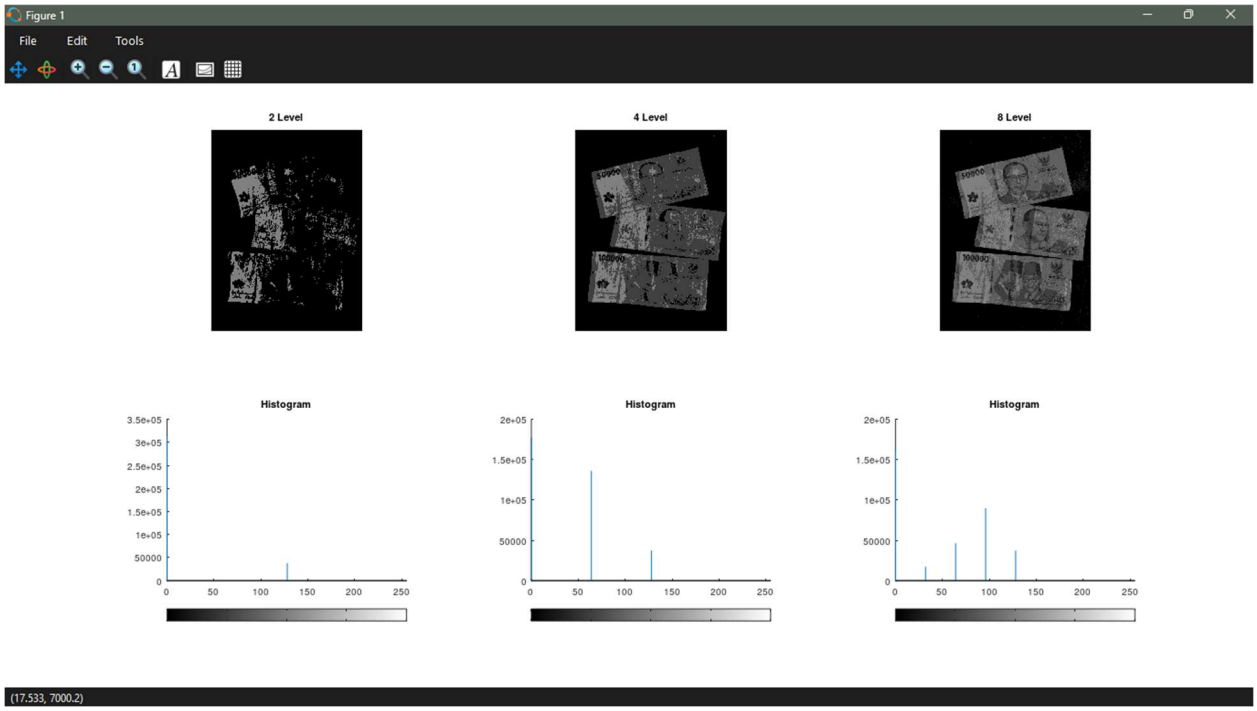
```
img = imread('E:\coolyeah\pengolahan citra\gambar\citraRGB.jpeg');
imshow (img); title ('Citra RGB');
```

d. Dalam pemrosesan keamera keamanan seringkali dilakukan menggunakan citra grayscale ataupun citra biner. Alasan utamannya adala efisien, Karenna proses yang dilakukan terhadap citra dua dimensi lebih ringan daripada citra tiga dimensi. Citra dua dimensi juga lebih mudah mendeeksi objek dengan memisahkan antara tepian objek dan latar belakang.

2. Kuantisasi citra

a.

Code :



```
pkg load image;

img = imread('E:\coolyeah\pengolahan citra\gambar\citraRGB.jpeg');
gray = rgb2gray(img);

level_list = [2, 4, 8];

figure;
for i = 1:length(level_list);
jumlah_level = level_list(i);
interval = 256 / jumlah_level;
kuantisasi = floor(double(gray) / interval) * interval;
kuantisasi = uint8(kuantisasi);

subplot(2, 3, i);
imshow(kuantisasi);
title([num2str(jumlah_level), ' Level']);
subplot (2,3,i+3);
imhist(kuantisasi);
title('Histogram');
end
```

b. Perbandingan Kuantisasi
   - Kuantisai dengan 2 level menghasilkan citra yang ga terlalu keliatan detailya. Karena kuantisasi 2 level hanya menghasilkan dua buah intensitas pixel dari seluruh citra.
   - Kuantisasi 4 level menghasilkan citra yang lebih jelas pada tepian objek. Tapi objekitu sendiri tidak terlihat jelas.
   - Kuantisasi 8 level menghasilkan citra yang detail, karena untensitas pixel yang terdapat dalam citra tersebut lebih banyak, menjadikan objek terlihat lebih detail.

c. Dalam pengiriman gambar melalui Whatsapp, kuantisasi bisa membantu memperkecil ukuran gambar dengan mengurangi intensitas pixel yang dimiliki, atau biasa disebut dengan kompresi gambar. Sehingga menghasilkan gambar denagan ukuran yang lebih ringan dan mempercepat proses transfer data gambar.

3. Kecerahan citra

a.

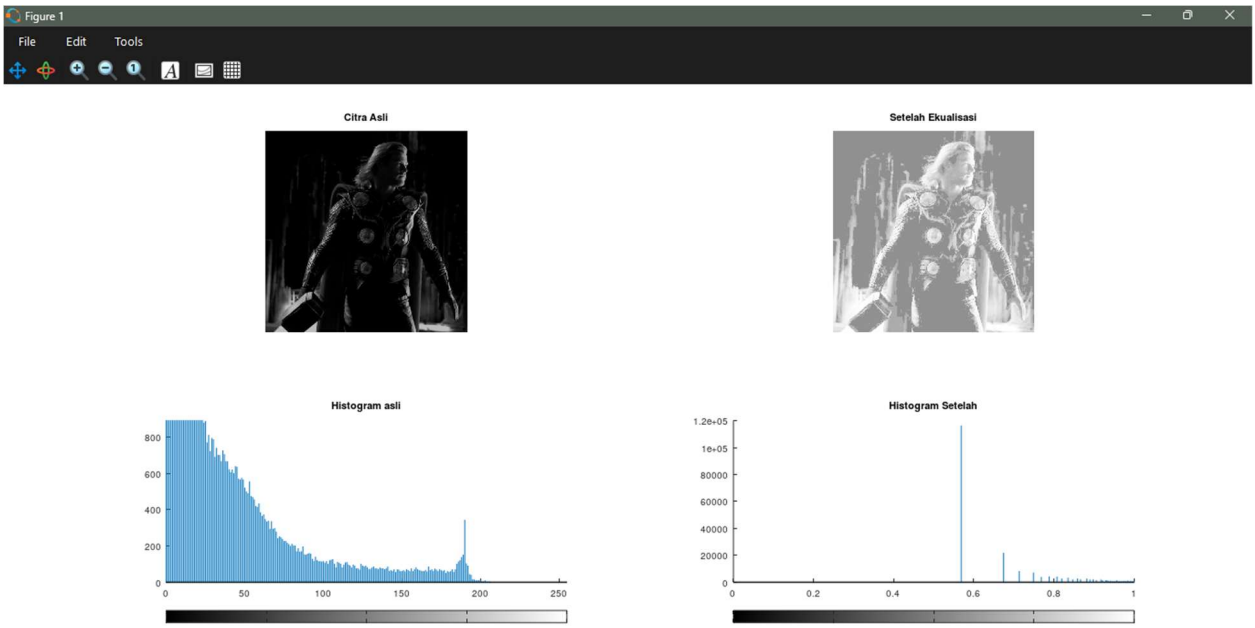
Code :

```
img = imread('E:\coolyeah\pengolahan citra\gambar\citraRGB.jpeg');
rgb = img + 50;
figure;
subplot(2,2,1); imshow (img); title('citra asli');
subplot(2,2,3); imshow (rgb); title('kecerahan ditingkatkan');
subplot(2,2,2); imhist (img); title('citra asli');
subplot(2,2,4); imhist (rgb); title('kecerahan ditingkatkan');
```

b. Objek yang awalnya kurang terlihat menjadi lebih jelas. tapi hanya sebatas kecerahan yang meningkat sehingga tidak menambah detai objek.

4. Ekualisasi citra

a.



Code :

```
pkg load image;
img = imread('E:\coolyeah\pengolahan citra\gambar\thor.jpeg');
gray = rgb2gray(img);
% Ekualisasi histogram
equalized = histeq(gray);
% Tampilkan perbandingan
subplot(2,2,1); imshow(gray); title('Citra Asli');
subplot(2,2,2); imshow(equalized); title('Setelah Ekualisasi');
subplot(2,2,3); imhist(gray); title('Histogram asli');
subplot(2,2,4); imhist(equalized); title('Histogram Setelah');
```

b. Equalisasi histogram sering dipakai dalam system pengenalan wajah karena ekualisasi meratakan pencahayaan pada citra, sehingga bagian bagian pada wajah seperti hidung, mata, mulut, bisa terlihat lebih jelas. seperti contoh perbandingan citra diatas, citra yang telah diekualisasi terlihat lebih jelas daripada citra aslinya.

5. Noise
   a.

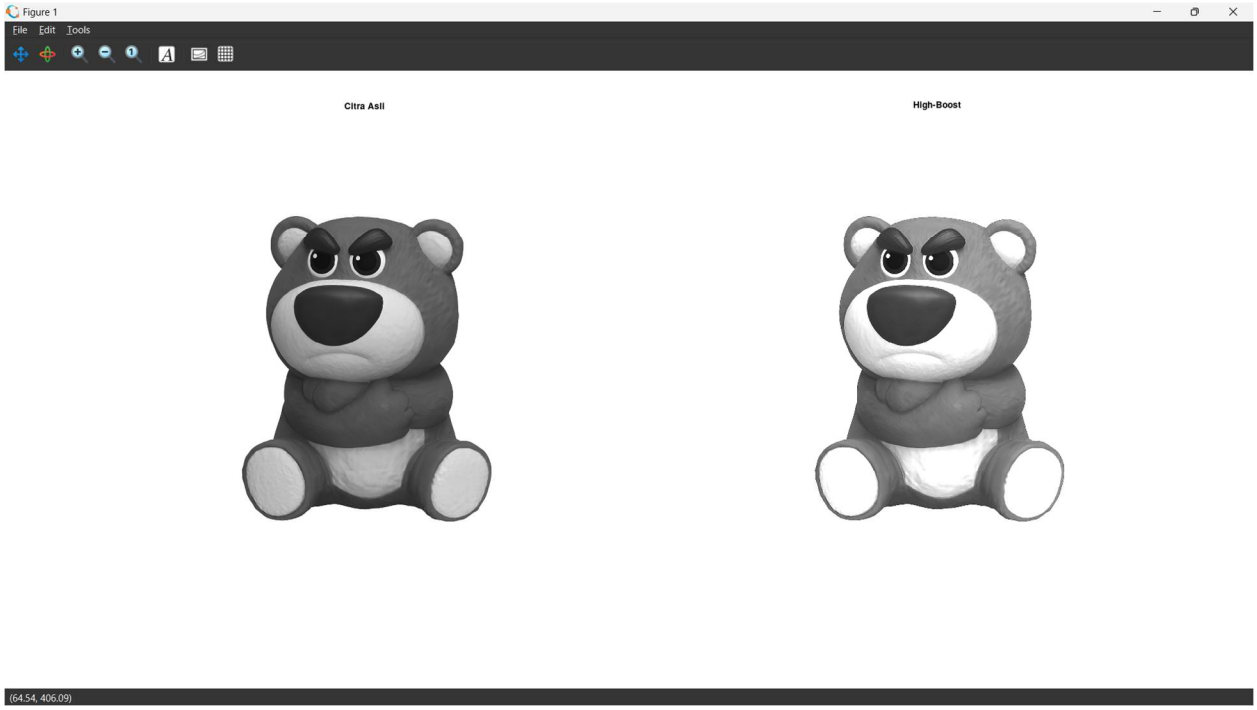
Code :

```
pkg load image;
%filter median
F = imread('E:\coolyeah\pengolahan citra\gambar\noise.jpg');
if size(F, 3) == 3
F = rgb2gray(F);
end
filtered_image = medfilt2(F, [3 3]);

% Filter pemerataan
kernel = ones(3, 3) / 9;
filtered_image = imfilter(F, kernel, 'symmetric');

figure;
subplot(1,3,1); imshow(F); title('Citra asli');
subplot(1,3,2); imshow(uint8(filtered_image)); title('Filter Rata rata');
subplot(1,3,3); imshow(filtered_image); title('Filter Median');
```

   b. Filter median menghasilkan citra yang lebih detail terhadap objek, namun bitnik bitnik nois marih terlihat dan tekstur lebih kasar. Sedangkan filter mean memberikan tekstur yang lebih halus dengan nois yang minim, tapi detail objek menjadi kurang.
   c. Berdasarkan percobaan tersebut filter median menghasilkan objek yang lebih detail walaupun masih menyisakan nois. Tapi di dunia medis detail dianggap lebih penting untuk mengetahui kondisi organ yang menjadi objek pemeriksaan.

6. High Boost

a.



Code :

```
F = imread('E:\coolyeah\pengolahan citra\gambar\lotso.jpg');

F = rgb2gray(F);
% Kernel Gaussian separabel
Hkol = [1 2 1]/4;
Hbrs = [1 2 1]/4;

% Faktor penguat high-boost
A = 2.5;

%Fungsi High-Boost
 [tinggi_f, lebar_f] = size(F);
 [~, lebar_h] = size(Hbrs);
m2 = floor(lebar_h/2);
F2 = double(F);
T = F2;

% Konvolusi vertikal
for y = m2+1 : tinggi_f - m2
for x = 1 : lebar_f
jum = 0;
for p = -m2 : m2
jum = jum + Hkol(p + m2 + 1) * F2(y - p, x);
end
T(y, x) = jum;
end
```

```
end

% Konvolusi horizontal (low-pass)
LPF = zeros(size(F2));
for y = 1 : tinggi_f
for x = m2+1 : lebar_f - m2
jum = 0;
for p = -m2 : m2
jum = jum + Hbrs(p + m2 + 1) * T(y, x - p);
end
LPF(y, x) = jum;
end
end
% rumus High-Boost
G = A * F2 - LPF;
G = uint8(max(min(G, 255), 0));

% Tampilkan hasil
subplot(1,2,1); imshow(F); title('Citra Asli');
subplot(1,2,2); imshow(G); title('High-Boost');
```

b. Melihat dari hasil percobaan diatas, filter high-boost menjadikan gambar lebih tajam dan detail yang lebih tinggi. Pemrosesan Optical Character Recognition menjadi lebih akurat dengan detail yang tinggi.

7. Pergrseran citra

a.

Code :

```
clc;
berkas =' E:\coolyeah\pengolahan citra\gambar\laptp.jpeg';
```

```
F = imread(berkas);
    img = rgb2gray(F);
F = double(img);

function G = tbilin(F, a1, a2, a3, a4, b1, b2, b3, b4)
 [tinggi, lebar] = size(F);
G = zeros(tinggi, lebar);  % Inisialisasi output

for y = 1 : tinggi
for x = 1 : lebar
x2 = a1 * x + a2 * y + a3 * x * y + a4;
y2 = b1 * x + b2 * y + b3 * x * y + b4;

if (x2 >= 1) && (x2 <= lebar - 1) && (y2 >= 1) && (y2 <= tinggi - 1)
p = floor(y2);
q = floor(x2);
a = y2 - p;
b = x2 - q;

% Interpolasi bilinear
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;
else
G(y, x) = 0;
end
end
end

G = uint8(G);
end
%acuan setting
##a1 = besarin gambar
##a2 = miring kanan kiri
##a3 = melengkkung horizontal
##a4 = geser kanan kiri (pixel)
##b1 = miring atas bawah
##b2 = memanjang keatas
##b3 = melengkung vertikal
##b4 = geser atas bawah (pixel)
% fungsi bilinier
G = tbilin(F, 0.3,0,0,140, 0,1,0,0);  % fungsi biliniear
figure;
subplot (1,2,1); imshow (G); title ('Biliniear');
subplot (1,2,2); imshow (img); title ('Gambar Asli');
```

b. Penangkapan gambar terhadap sebuah dokumen tidak selalu presisi dari sudut yang tepat, maka dibutuhkan penesuaian terhadap citra digital dokumen untuk menghasilkan citra dengan bentuk yang ideal.

8. Efek

a.

Code :

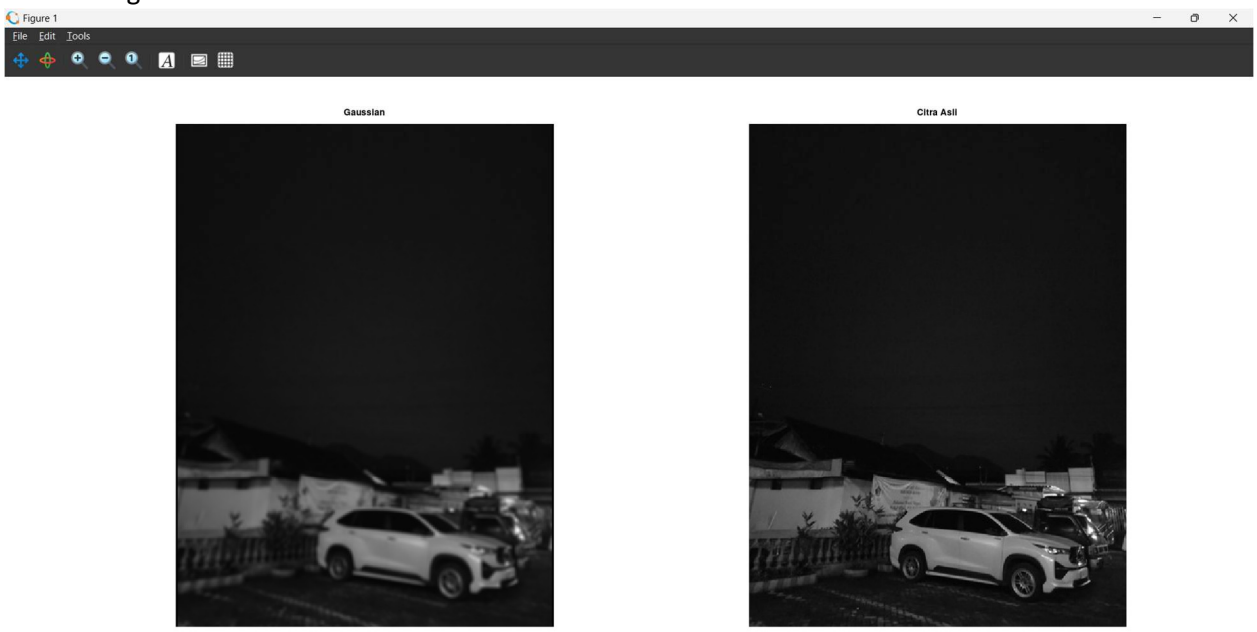
```
clc;
F = imread('D:\rizalDisini\pengolahan citra\gambar\lotso.jpg');
function G = ripple(F, ax, ay, tx, ty)
% RIPPLE Berfungsi untuk melakukan transformasi 'ripple'.
dimensi = size(F);
tinggi = dimensi(1);
lebar = dimensi(2);
for y=1 : tinggi
for x=1 : lebar
x2 = x + ax * sin(2 * pi * y / tx);
y2 = y + ay * sin(2 * pi * x / ty);
if (x2>=1) && (x2<=lebar) && ...
(y2>=1) && (y2<=tinggi)
% Lakukan interpolasi bilinear
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 = (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;
```

```
end
else
G(y, x) = 0;
end
end
end
end
G = uint8(G);
end

G = ripple(F,10,15,120, 250);
subplot(1,2,1); imshow(G); title('Ripple');
subplot(1,2,2); imshow(F); title('Cirta Asli');
```

b. Efek ripple memberikkan hasil citra unik yang tidakbisa diambil secara langsung tanpa melalui pemrosesan citra.

9. Filter gaussian

a. Perbandingan



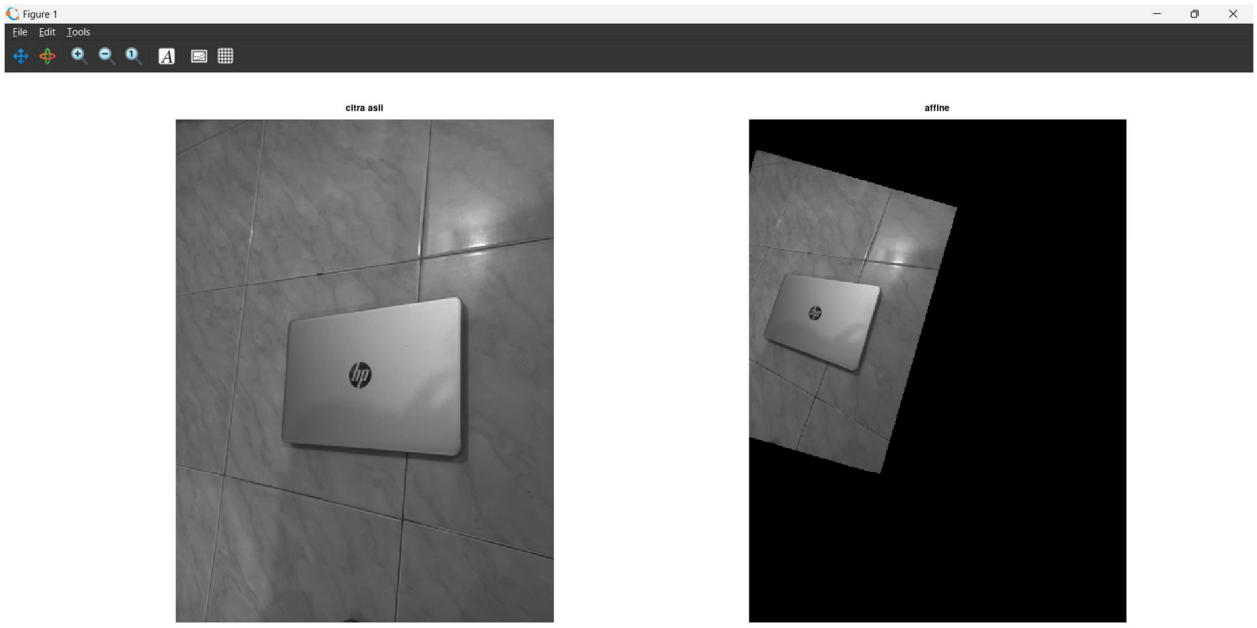
Code :

```
%F=imread ('D:\rizalDisini\pengolahan citra\gambar\park.jpeg');
F = imread('D:\rizalDisini\pengolahan citra\gambar\park.jpeg');
F = rgb2gray(F);

kernel_size = 5;  % ukuran kernel Gaussian harus ganjil
sigma = 2.0;   % standar deviasi Gaussian

function [G] = konvolusi_gaussian(F, kernel_size, sigma)

% kernel Gaussian 1D
m2 = floor(kernel_size / 2);
```

```
x = -m2:m2;
H = exp(-(x.^2) / (2*sigma^2));
H = H / sum(H);  % Normalisasi

Hkol = H;  % Filter vertikal (1D horizontal)
Hbrs = H;  % Filter horizontal (1D horizontal)

[tinggi_f, lebar_f] = size(F);
F2 = double(F);
T = F2;

% Konvolusi vertikal dengan Hkol
for y = m2+1 : tinggi_f - m2
for x = 1 : lebar_f
jum = 0;
for p = -m2 : m2
jum = jum + Hkol(p + m2 + 1) * F2(y - p, x);
end
T(y, x) = jum;
end
end

% Konvolusi horizontal dengan Hbrs
G = zeros(size(F2)); % Inisialisasi output
for y = 1 : tinggi_f
for x = m2+1 : lebar_f - m2
jum = 0;
for p = -m2 : m2
jum = jum + Hbrs(p + m2 + 1) * T(y, x - p);
end
G(y, x) = jum;
end
end

G = uint8(G);
end

G = konvolusi_gaussian(F, kernel_size, sigma);
subplot(1,2,1); imshow(G); title('Gaussian');
subplot(1,2,2); imshow(F); title('Citra Asli');
```

b. Filter gaussian mengkaburkan citra asli sehingga menjadikan citra asli tidak terlihat jelas, sehingga bisa menyamarkan objek yang dianggap privasi.

10. transformasi
    a.



Code :

```
F = rgb2gray(imread('E:\coolyeah\pengolahan citra\gambar\laptp.jpeg'));

function G = taffine(F, a11, a12, a21, a22, tx, ty)

[tinggi, lebar] = size(F);
for y=1 : tinggi
for x=1 : lebar
x2 = a11 * x + a12 * y + tx;
y2 = a21 * x + a22 * y + ty;
if (x2>=1) && (x2<=lebar) && ...
(y2>=1) && (y2<=tinggi)
% interpolasi bilinear
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 = (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;
end
else
G(y, x) = 0;
end
```

```
end
end
G = uint8(G);
end

##parameter : G = taffine(F, a11, a12, a21, a22, tx, ty)
##a11, a22 = skala dan rotasi
##a12,a21 = rotasi dan shear
##tx, ty = translasi

rad = pi/6;
G = taffine(F, 2 * cos(rad) ,sin(rad) ,-sin(rad),  2 * cos(rad), -30,-
50);
subplot(1,2,1); imshow (F); title ('citra asli');
subplot(1,2,2); imshow (G); title ('affine');
```

b. Bangunan yang difoto dari atas menggunakan drone tidak selalu pada titik yang tepat, bisa saja miring atau distorsi yang dihasilkan kamera. Dengan menggunakan transformasi affine gambar bangunan bisa dikembalikan seperti aslinya.