TUGAS 3

IF4073 PEMROSESAN CITRA DIGITAL

Dibuat Oleh:

Christian Albert Hasiholan / 13521078

Tobias Natalio Sianipar / 13521090

PROGRAM STUDI TEKNIK INFORMATIKA

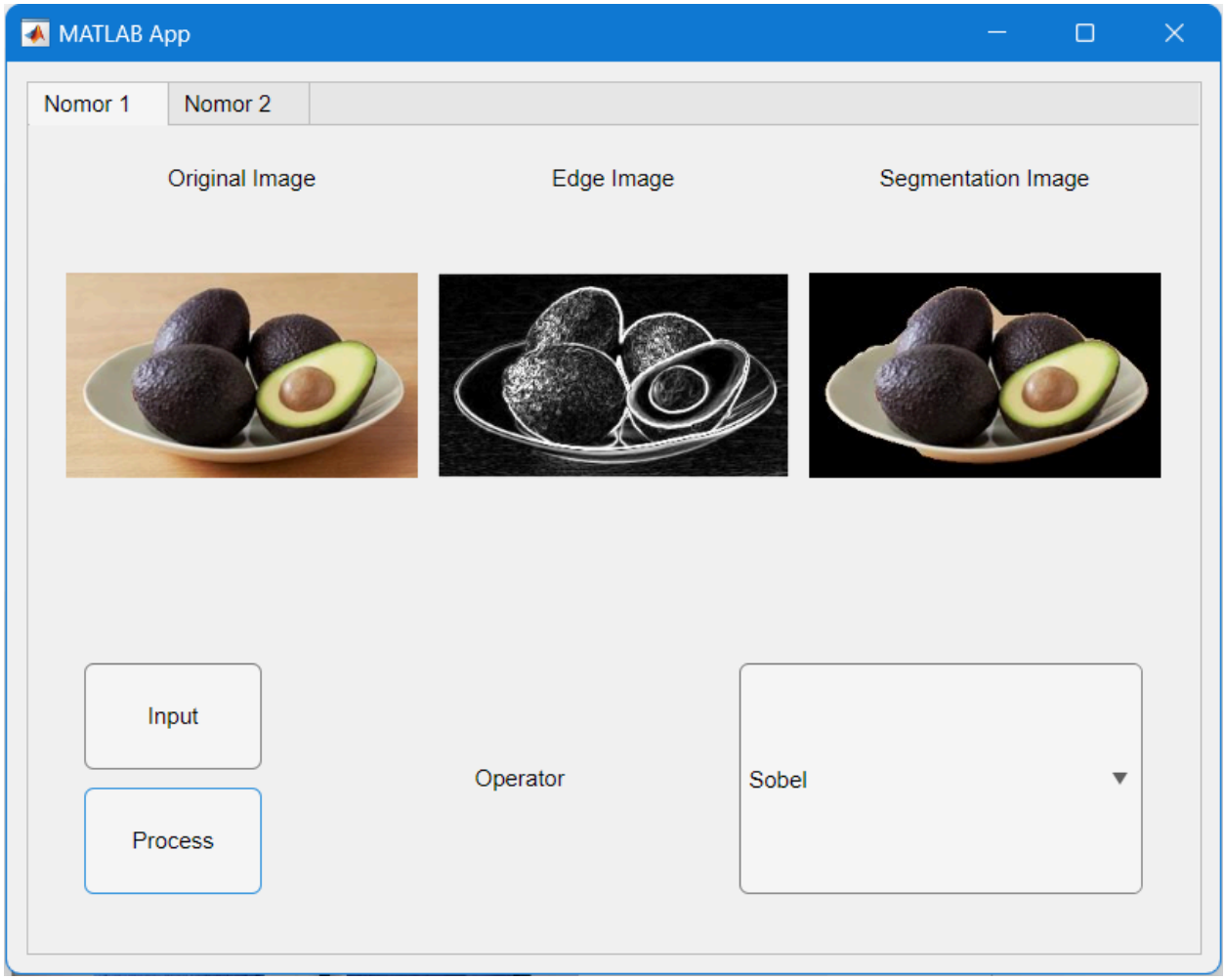
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

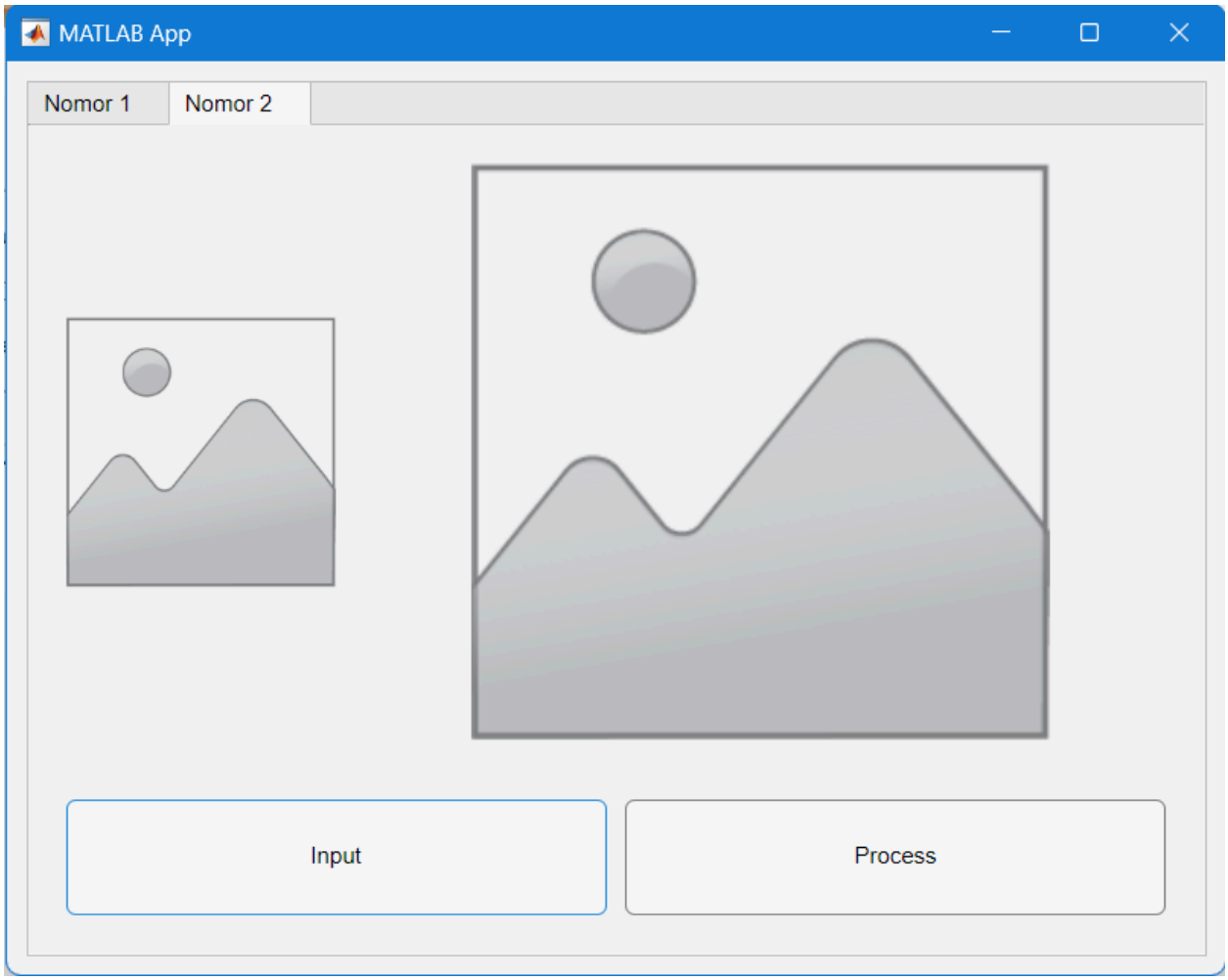
BANDUNG

1. GUI Program

Menu 1



Menu 2



2. Segmentasi Objek dengan Deteksi Tepi

2.1. Kode Program

```
function result = laplaceDetection(image)
    % Laplace mask
    H = [1 1 1; 1 -8 1; 1 1 1];
    I = im2gray(image);
    % Convolution with mask
    result = uint8(conv2(double(I), double(H)));

    % Remove extra pixels to match original size
    result = result(2:end-1, 2:end-1);
end
```

```
function result = logDetection(image)
    I = im2gray(image);
    % LoG mask
```

```
    h = fspecial('log');
    % Convolution with mask
    result = uint8(conv2(double(I), double(h)));
    % Remove extra pixels to match original size
    result = result(3:end-2, 3:end-2);
end
```

```
function result = sobelDetection(image)
    I = im2gray(image);
    % Sobel mask
    Sx = [-1 0 1; -2 0 2; -1 0 1];
    Sy = [1 2 1; 0 0 0; -1 -2 -1];
    % Convolution with mask
    Jx = conv2(double(I), double(Sx), 'same');
    Jy = conv2(double(I), double(Sy), 'same');
    % Build image edge from convolution result
    Jedge = sqrt(Jx.^2 + Jy.^2);
    result = uint8(Jedge);
end
```

```
function result = prewittDetection(image)
    I = im2gray(image);
    % Prewitt mask
    Sx = [-1 0 1; -1 0 1; -1 0 1];
    Sy = [-1 -1 -1; 0 0 0; 1 1 1];
    % Convolution with mask
    Jx = conv2(double(I), double(Sx), 'same');
    Jy = conv2(double(I), double(Sy), 'same');
    % Build image edge from convolution result
    Jedge = sqrt(Jx.^2 + Jy.^2);
    result = uint8(Jedge);
end
```

```
function result = robertsDetection(image)
    I = im2gray(image);
    % Roberts mask
    Rx = [1 0; 0 -1];
    Ry = [0 1; -1 0];
    % Convolution with mask
    Jx = conv2(double(I), double(Rx), 'same');
    Jy = conv2(double(I), double(Ry), 'same');
    % Build image edge from convolution result
    Jedge = sqrt(Jx.^2 + Jy.^2);
    result = uint8(Jedge);
end
```

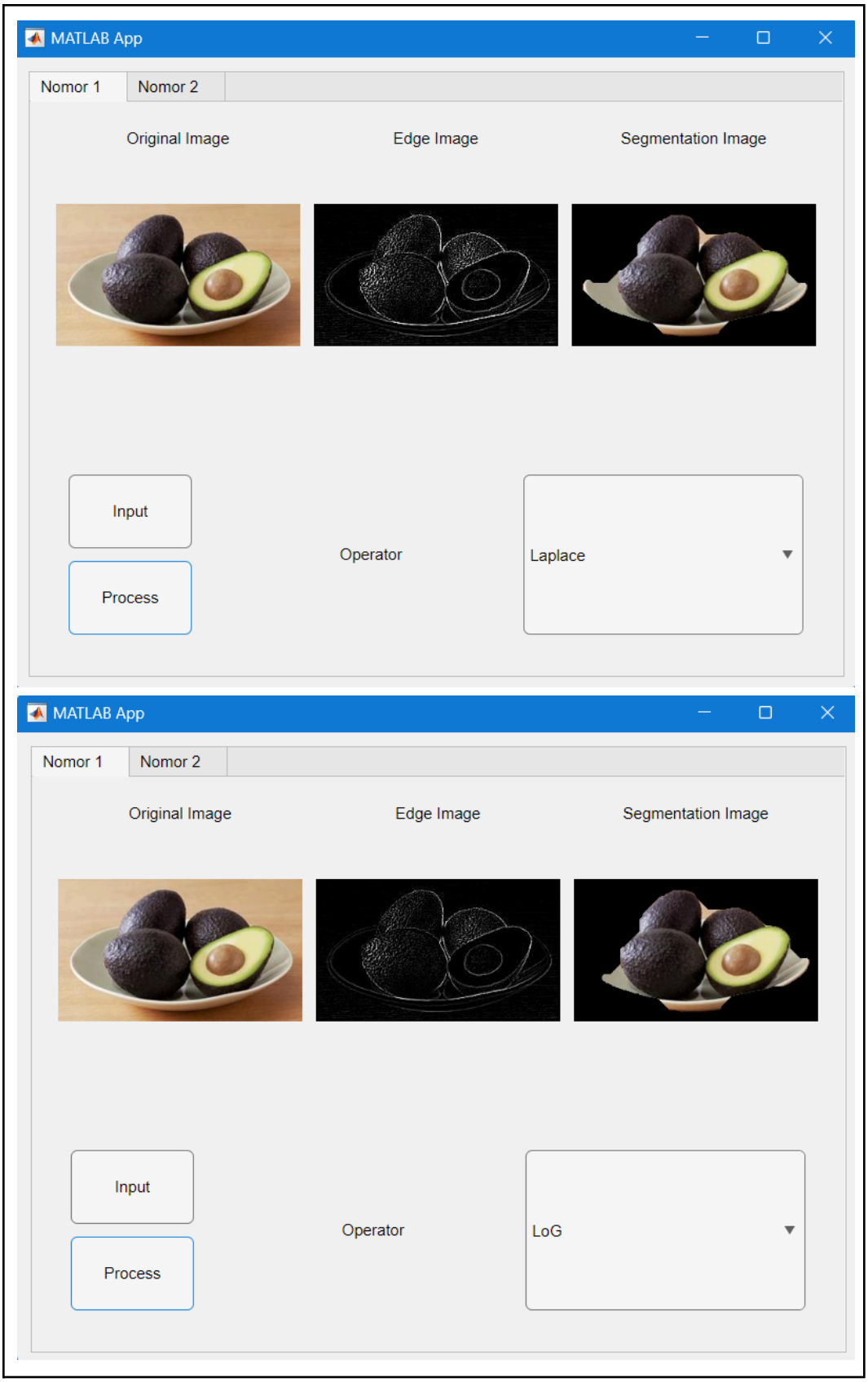
```
function result = cannyDetection(image)
    % Get image edge with default Canny
    image_edge = edge(im2gray(image), 'canny');
    % Convert image type
    result = uint8(image_edge) * 255;
end
```

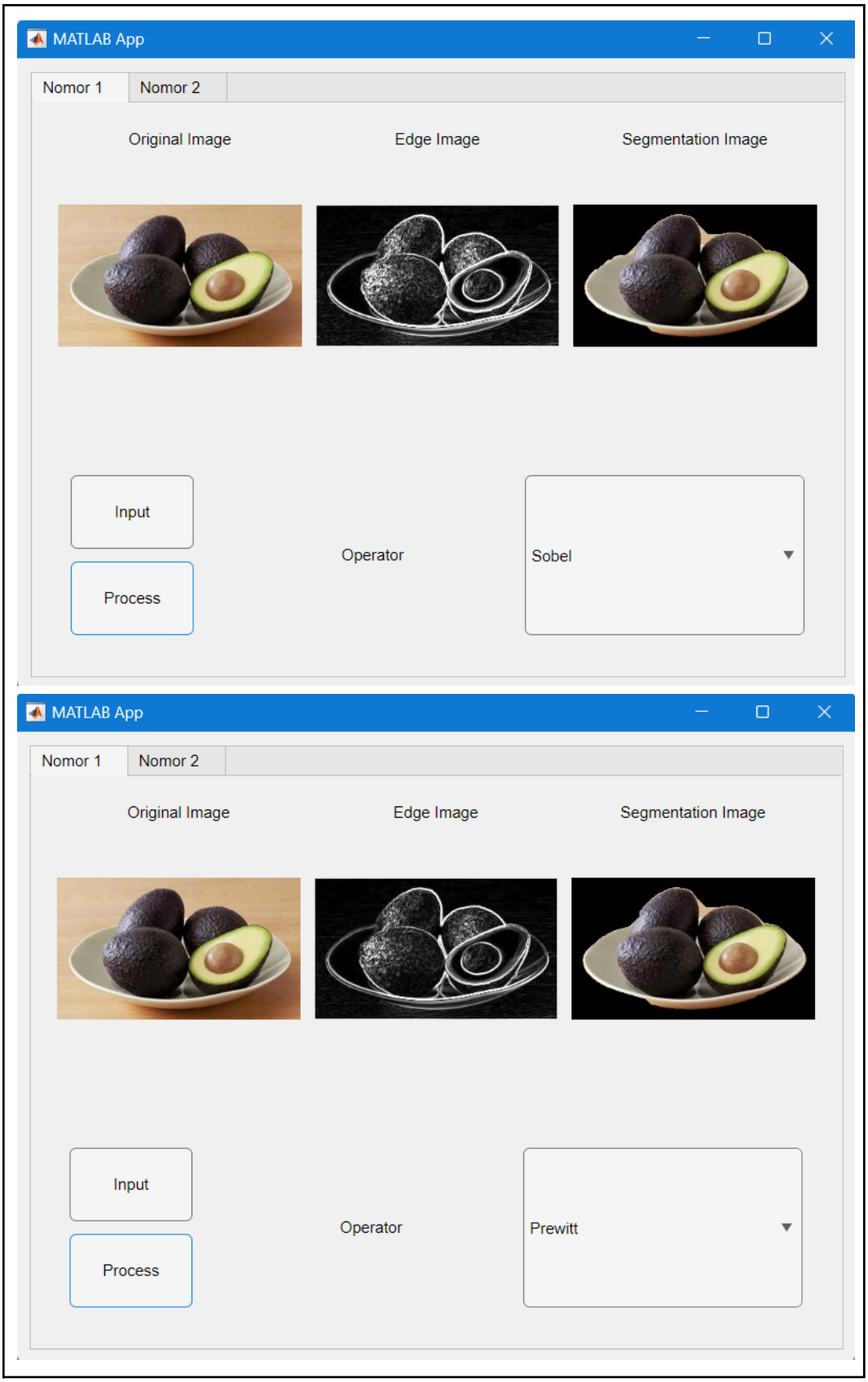
```
function result = objectSegmentation(image, image_edge)
    % Change border color to black
    image_edge(1, :, :) = 0;
```

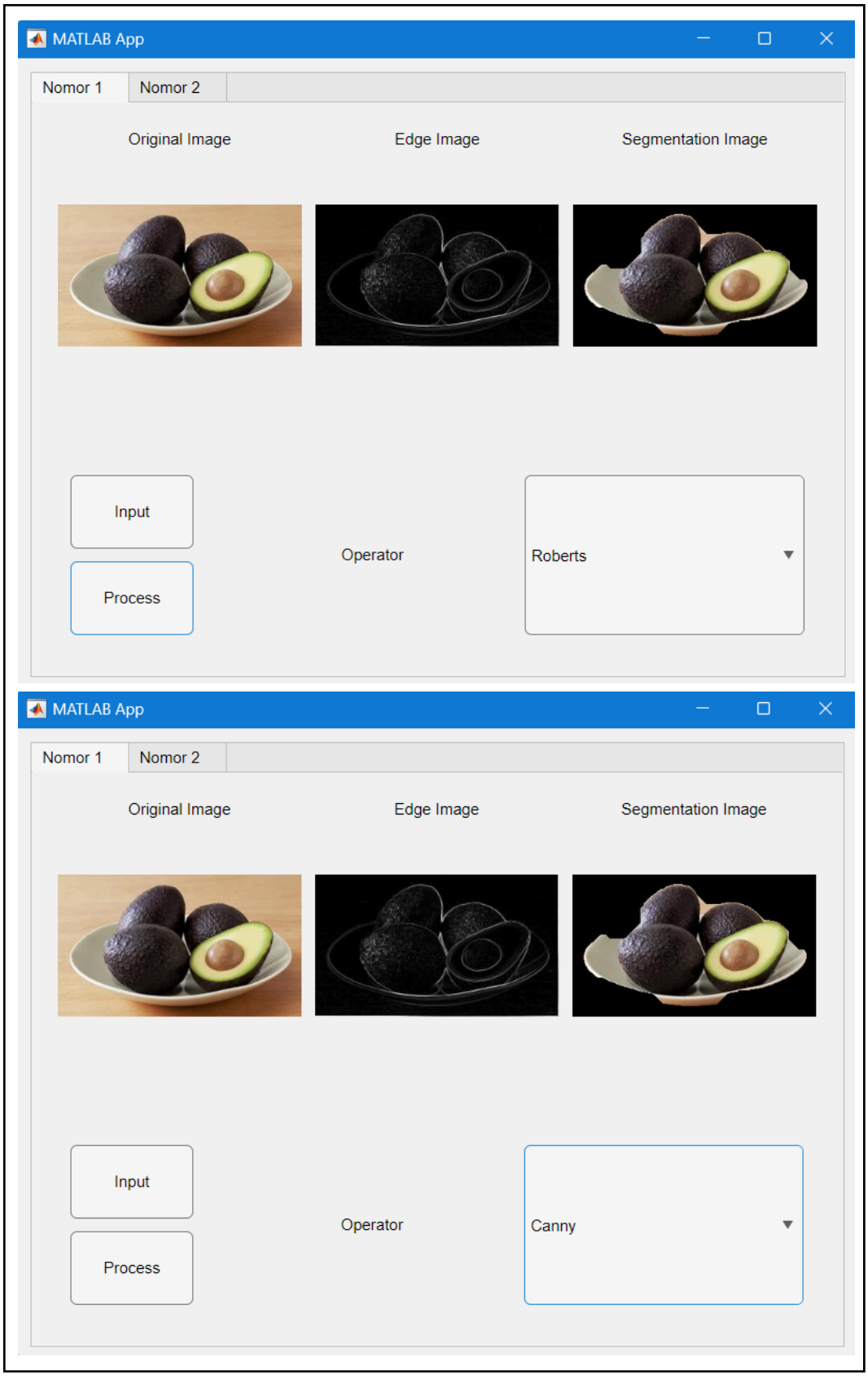
```
    image_edge(end, :, :) = 0;
    image_edge(:, 1, :) = 0;
    image_edge(:, end, :) = 0;

    % Change from grayscale to binary
    binarized_edge = imbinarize(image_edge);

    % Removing small pixel (not the object)
    binarized_edge = bwareaopen(binarized_edge, 5);
    binarized_edge = imopen(binarized_edge, ones(1, 1));

    % Close the edge of the object
    closed_edge = imclose(binarized_edge, strel('disk', 22));

    % Fill the object segment
    totalPixel = numel(closed_edge);
    whitePixel = sum(closed_edge(:));
    if (whitePixel/totalPixel) > 0.73
        filled_segment = imcomplement(closed_edge);
    else
        filled_segment = imfill(closed_edge, "holes");
    end

    % Remove small pixel again
    open_segment = imopen(filled_segment, strel('disk', 5));
    removed_bw = bwareaopen(open_segment, 1000);

    if size(image, 3) == 3
        red_channel = image(:,:,1).*uint8(removed_bw);
        green_channel = image(:,:,2).*uint8(removed_bw);
        blue_channel = image(:,:,3).*uint8(removed_bw);
        result = cat(3, red_channel, green_channel, blue_channel);
    else
        result = image.*uint8(removed_bw);
    end
end
```

2.2. Contoh Hasil Eksekusi

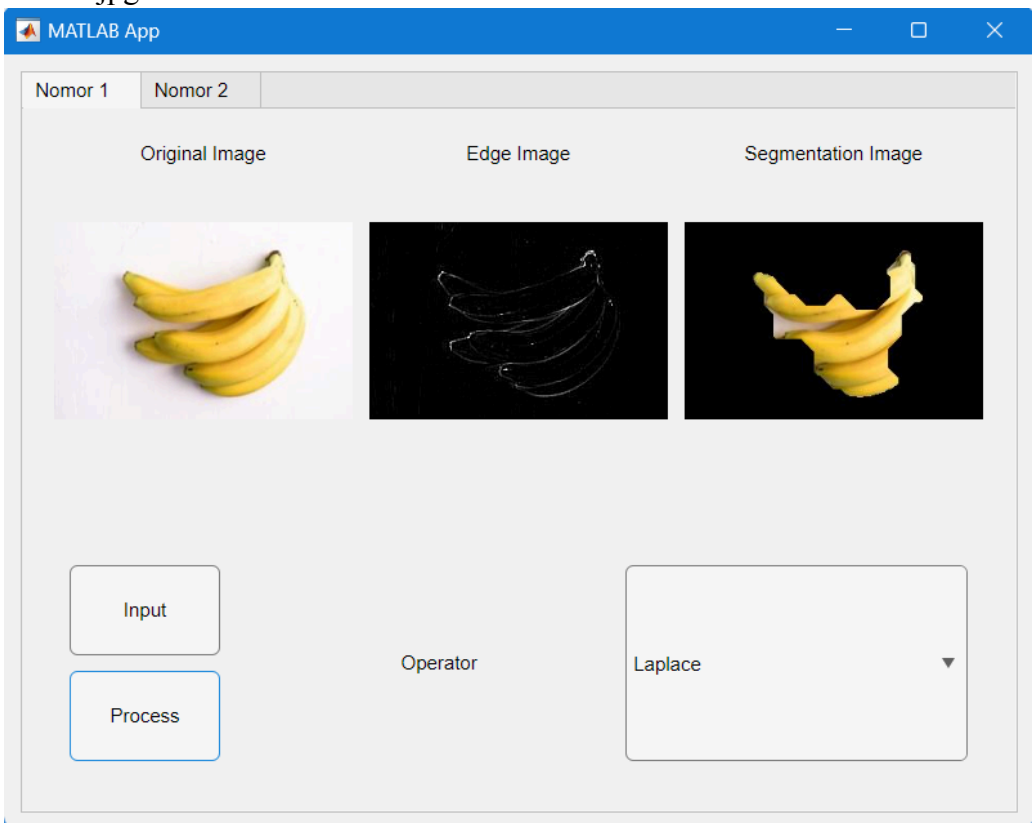
avocado.jpg

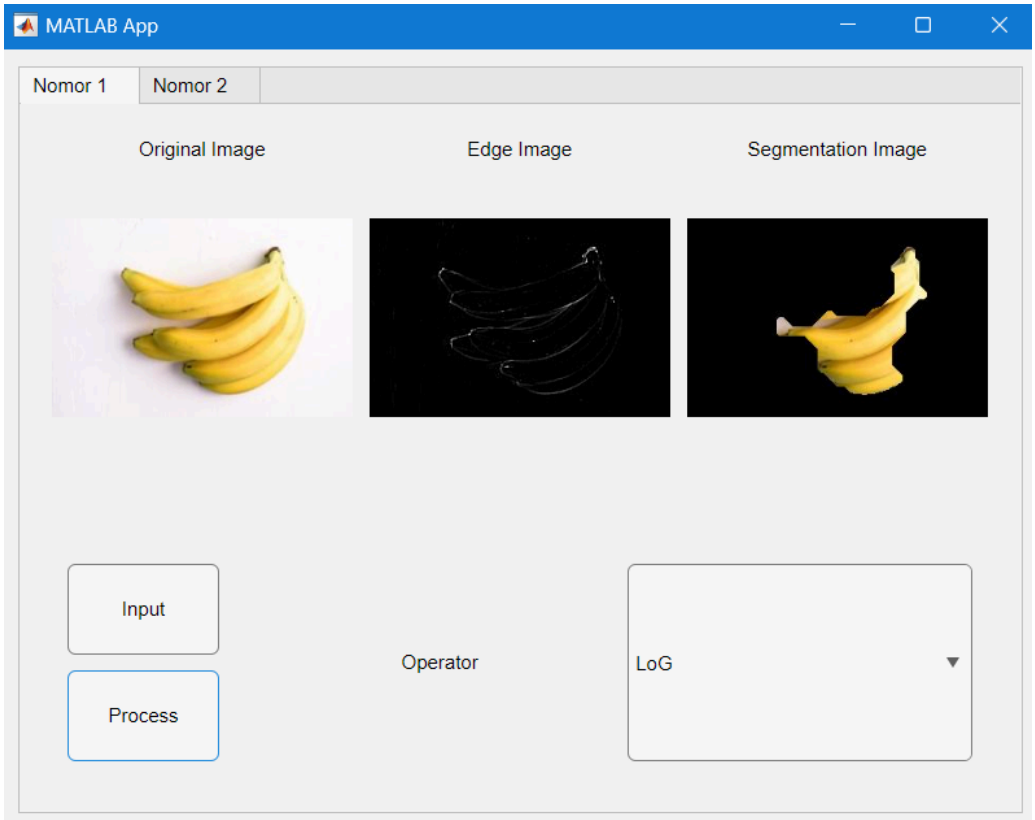

MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Laplace
MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
LoG

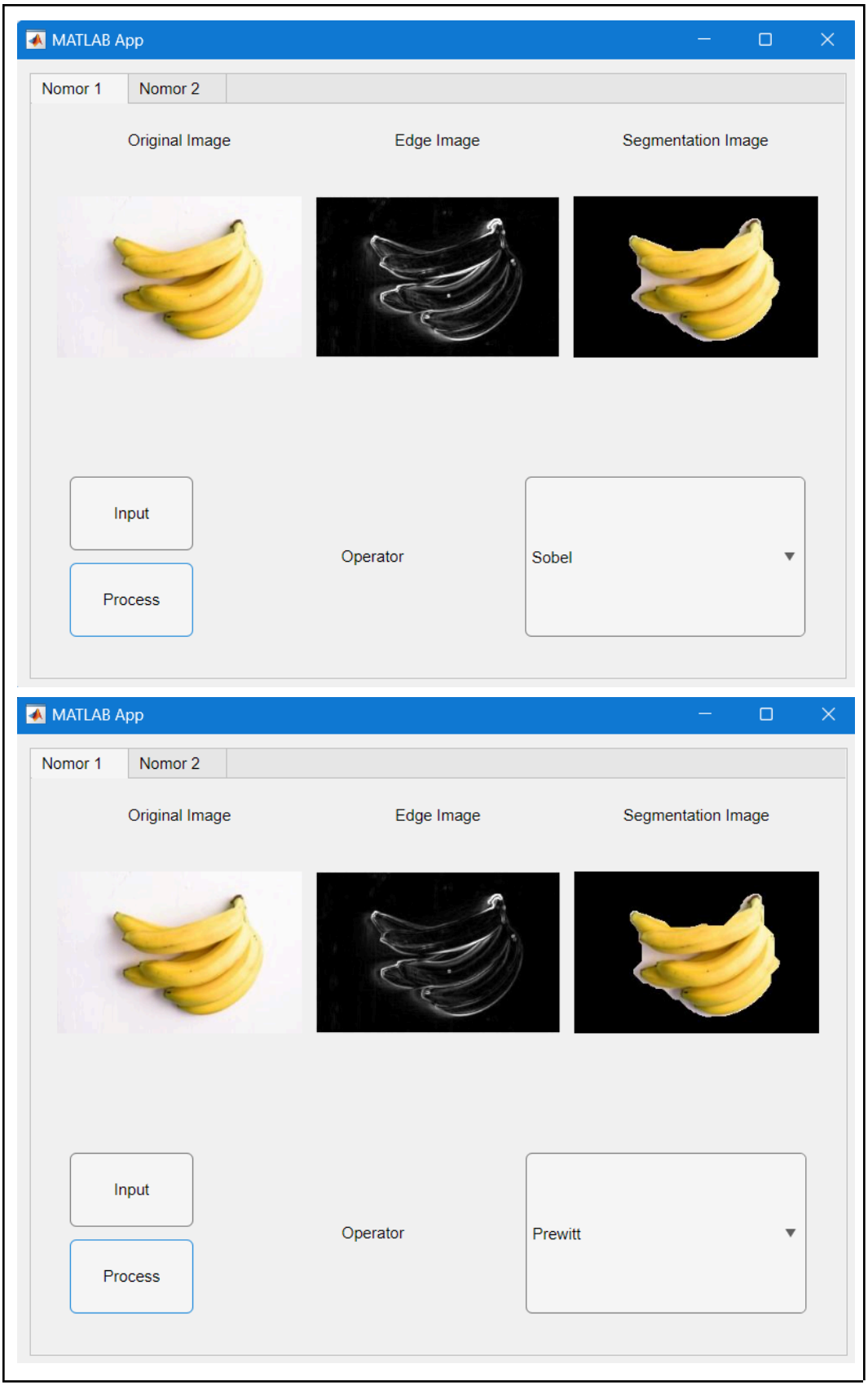

MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Sobel
MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Prewitt

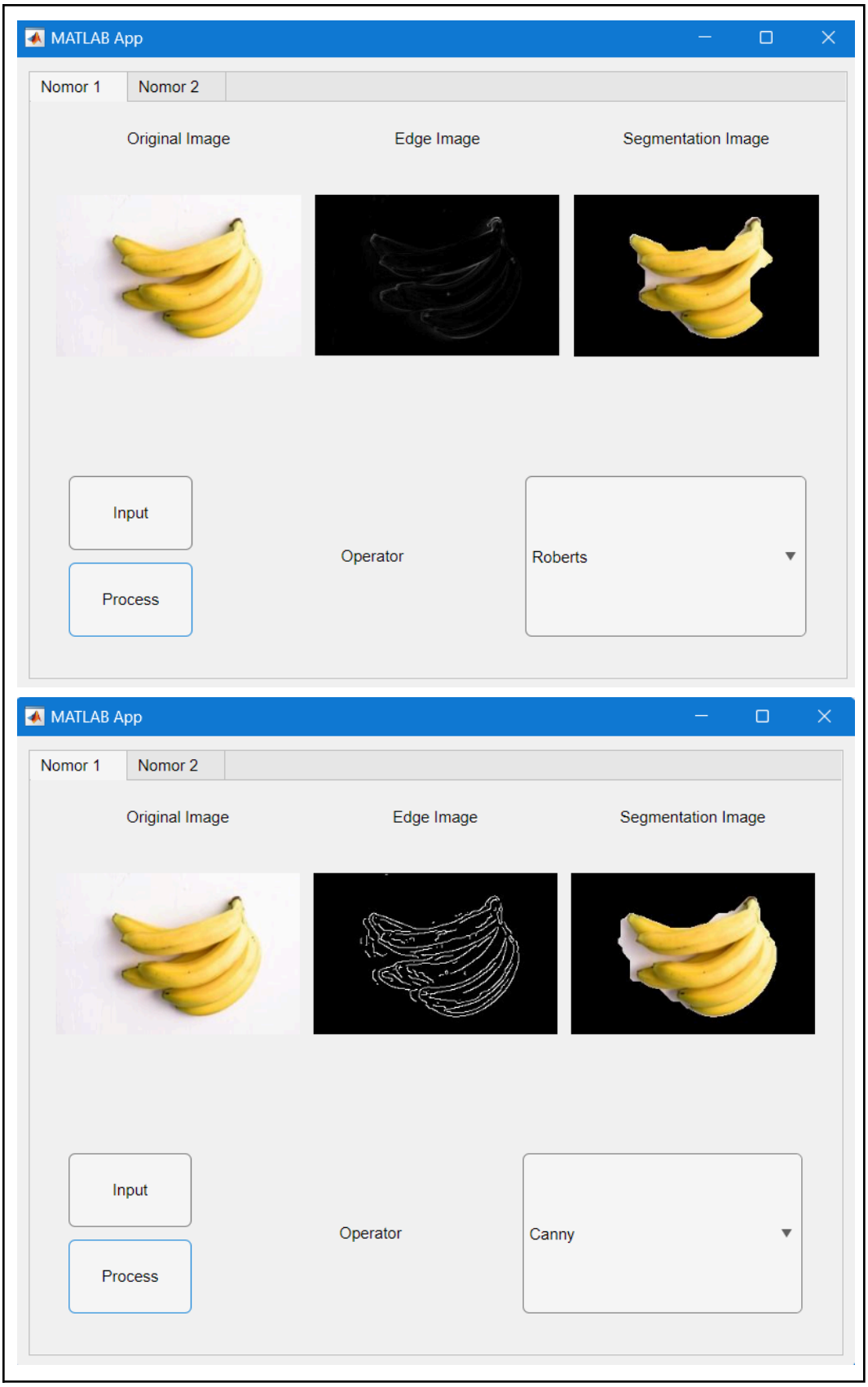

MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Roberts
MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Canny

banana.jpg

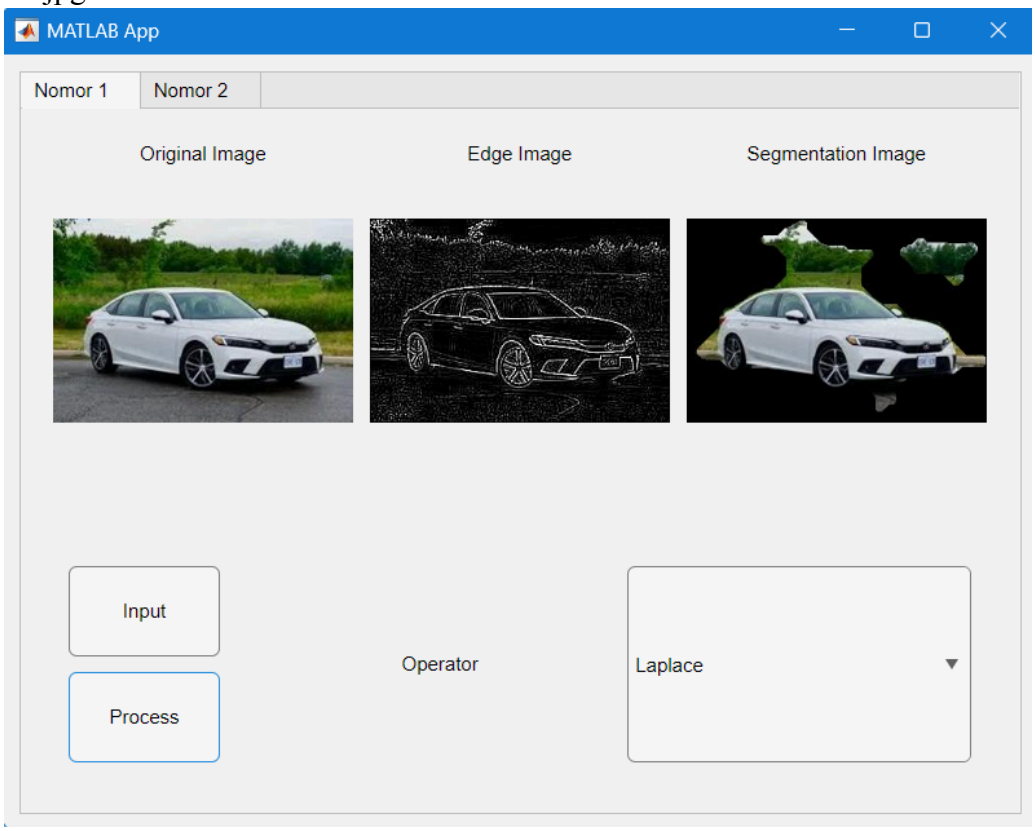

MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Laplace

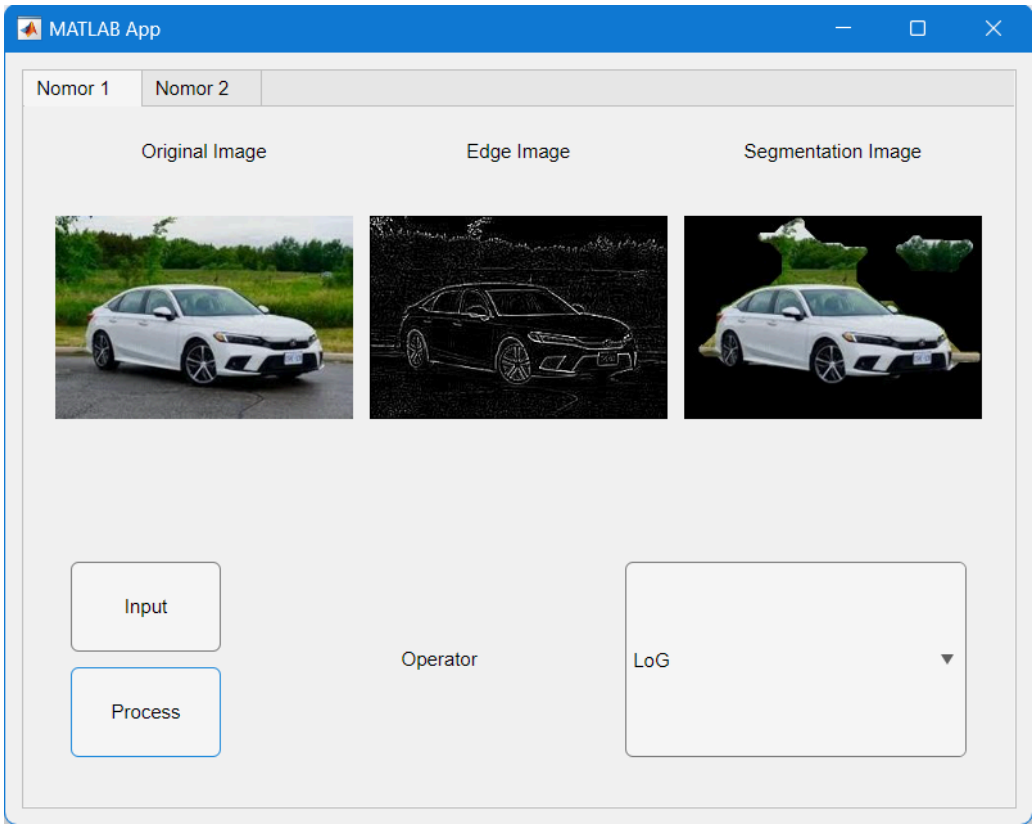

MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
LoG

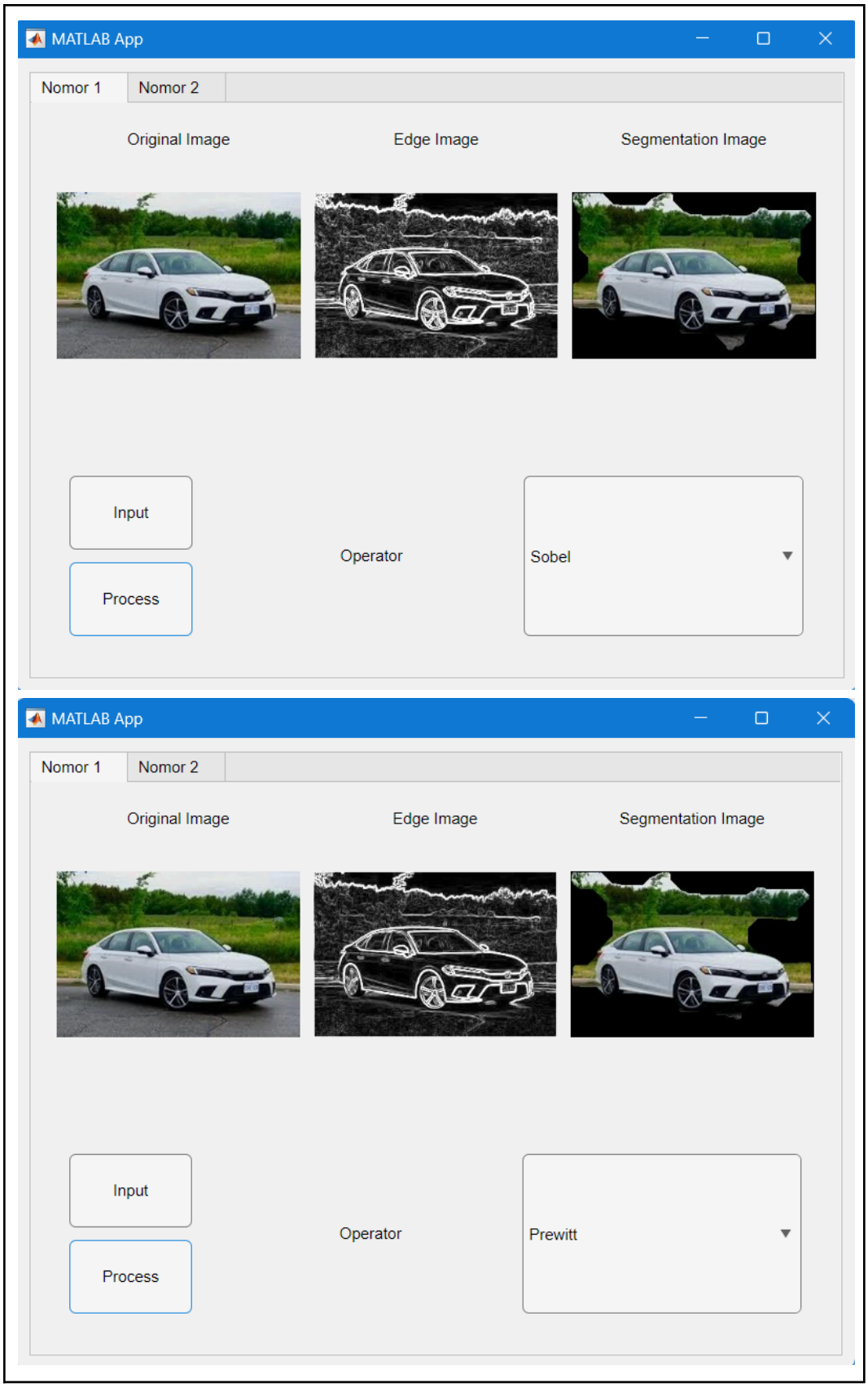

MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Sobel
MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Prewitt

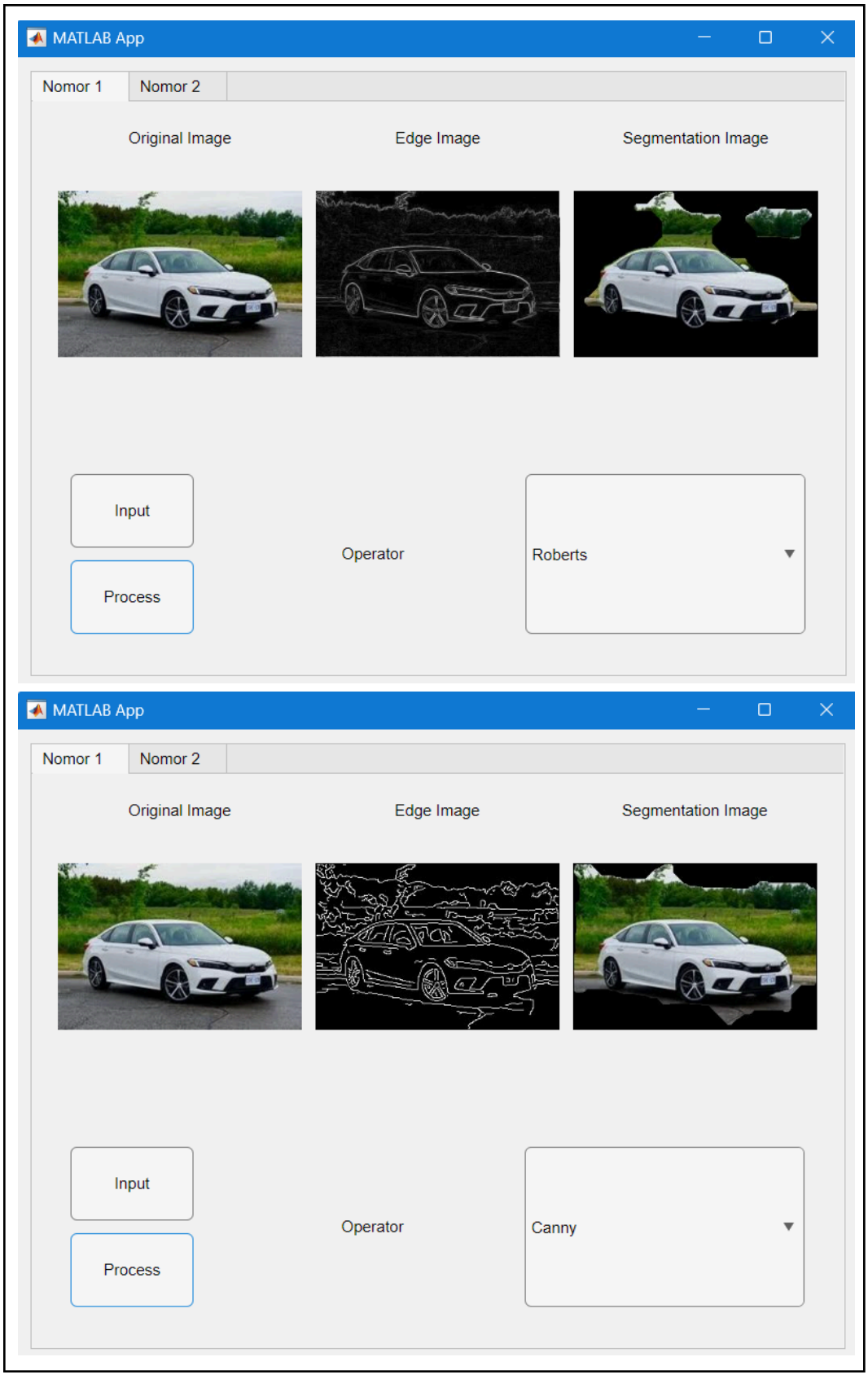

MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Roberts
MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Canny

car.jpg

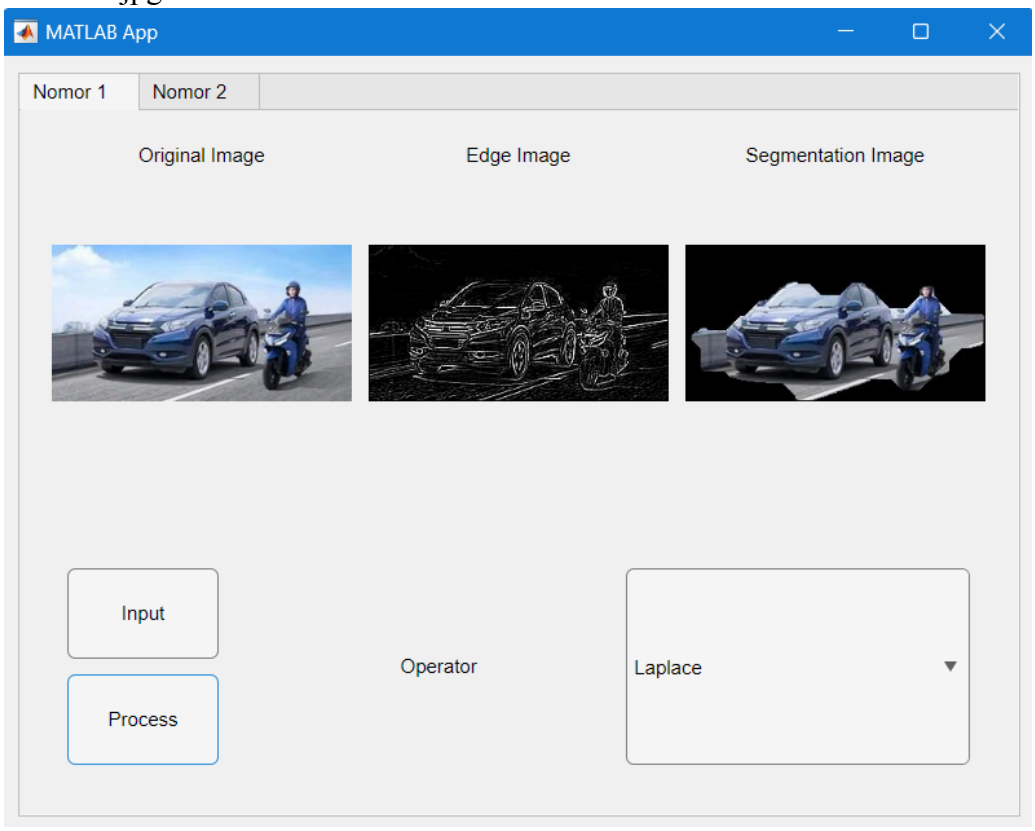

MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Laplace

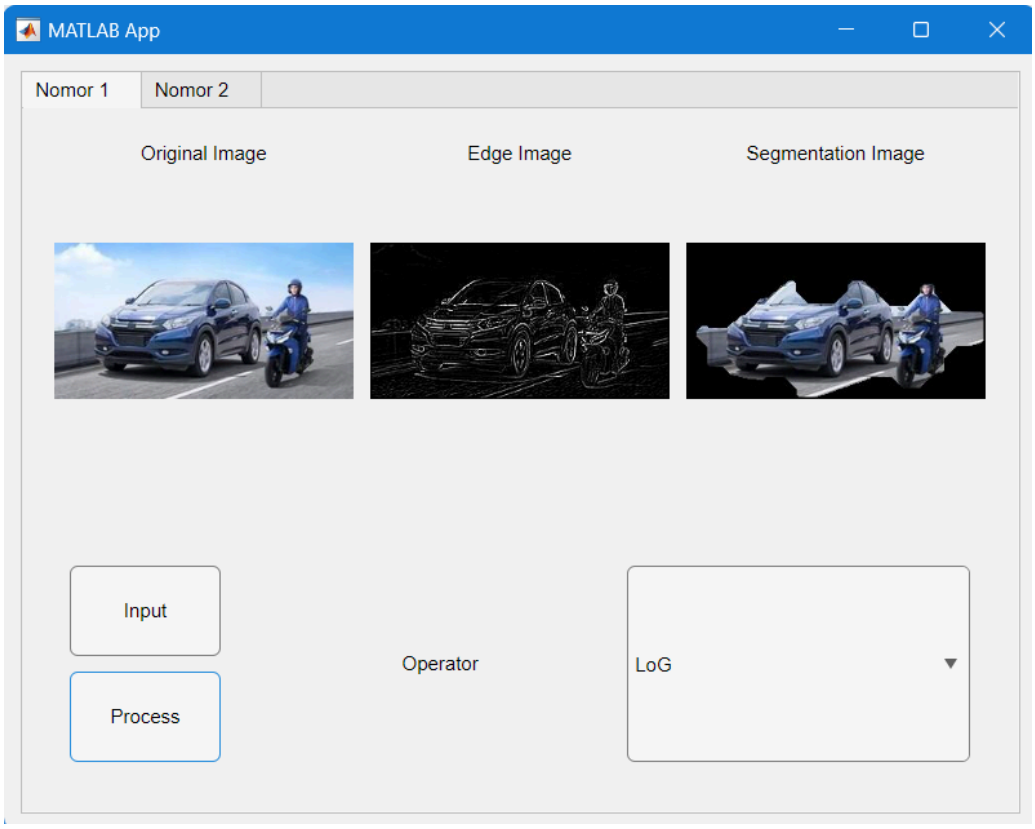

MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
LoG

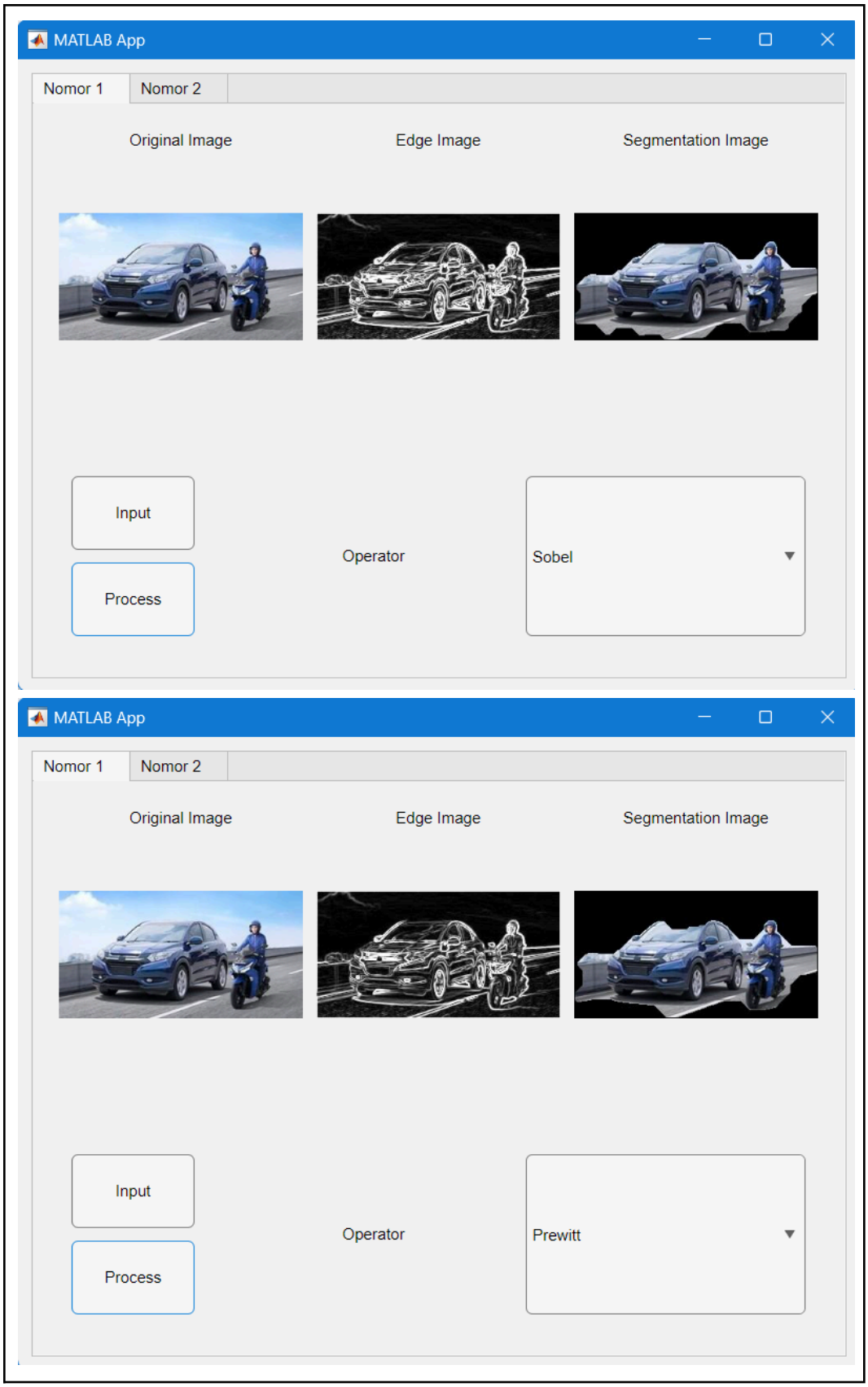

MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Sobel
MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Prewitt

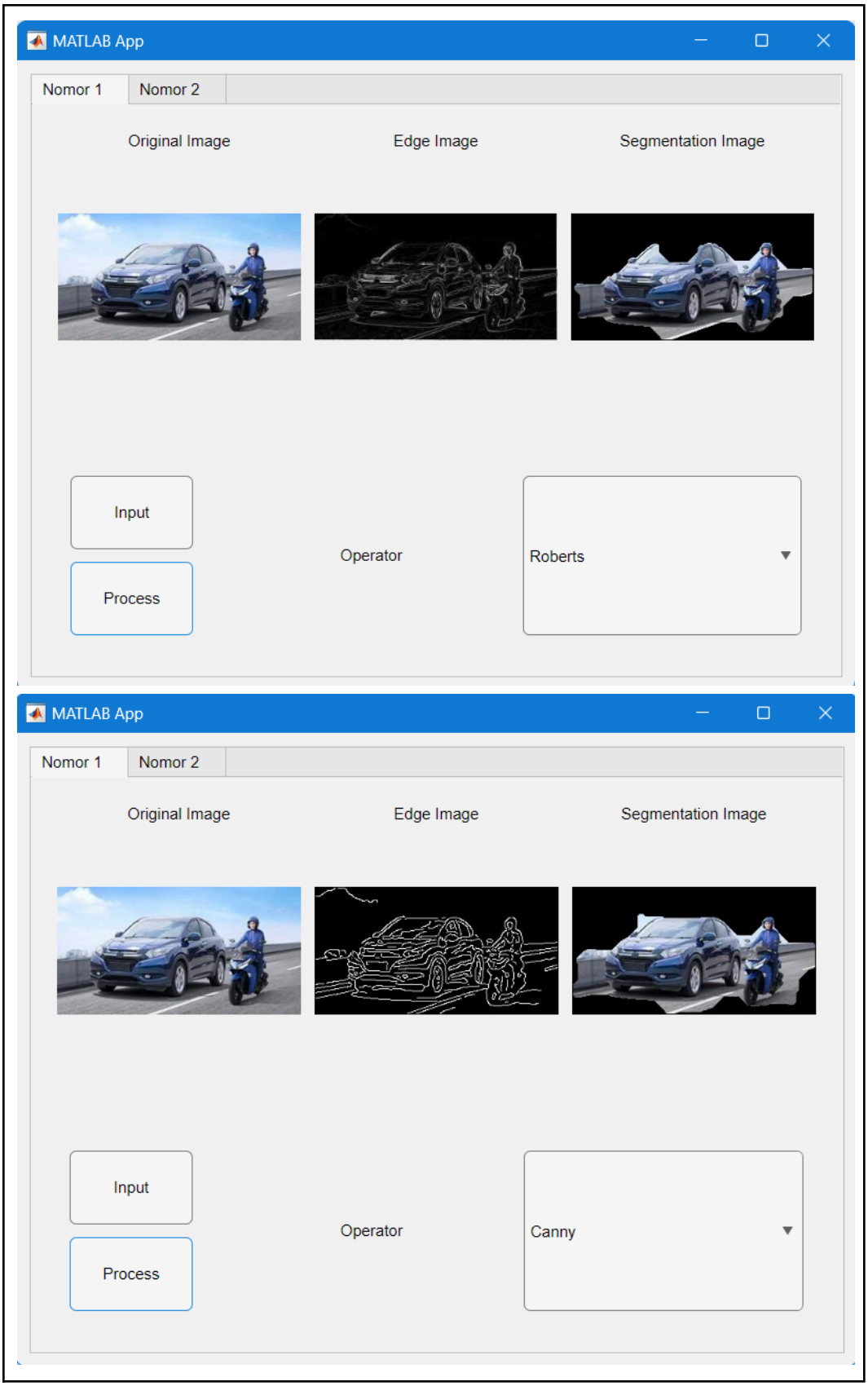

MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Roberts
MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Canny

carmotor.jpg

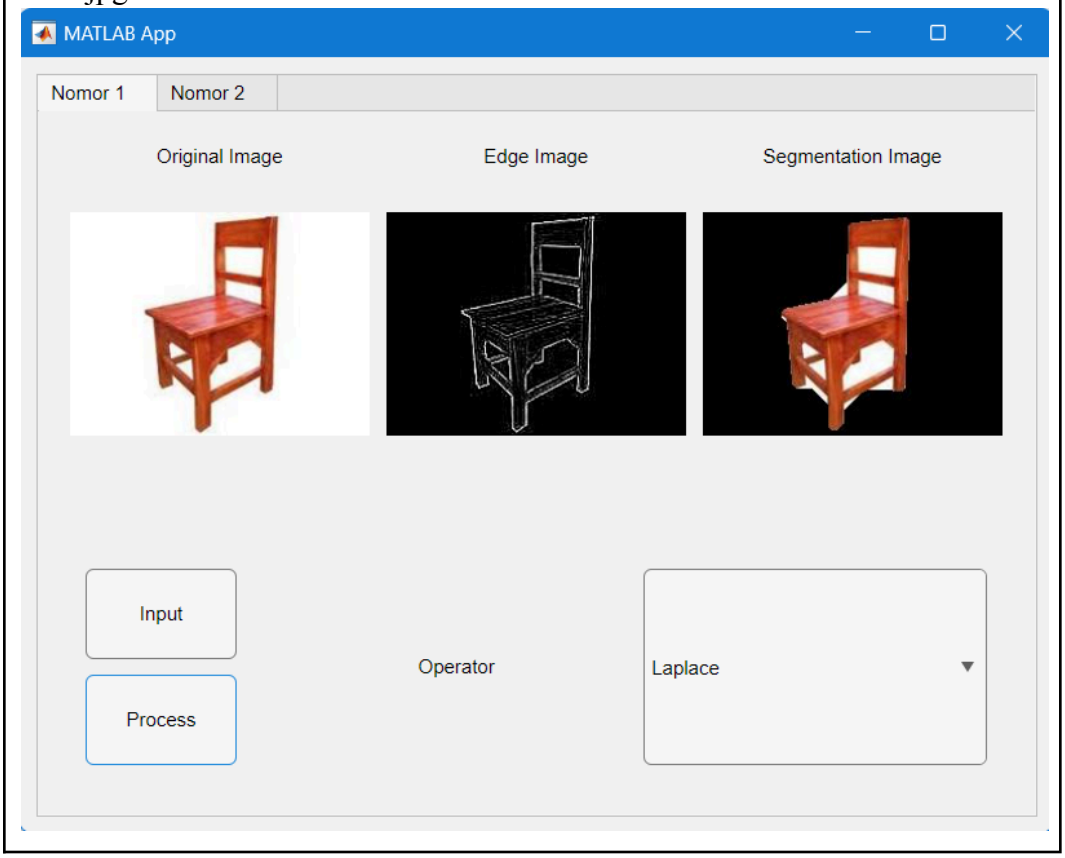


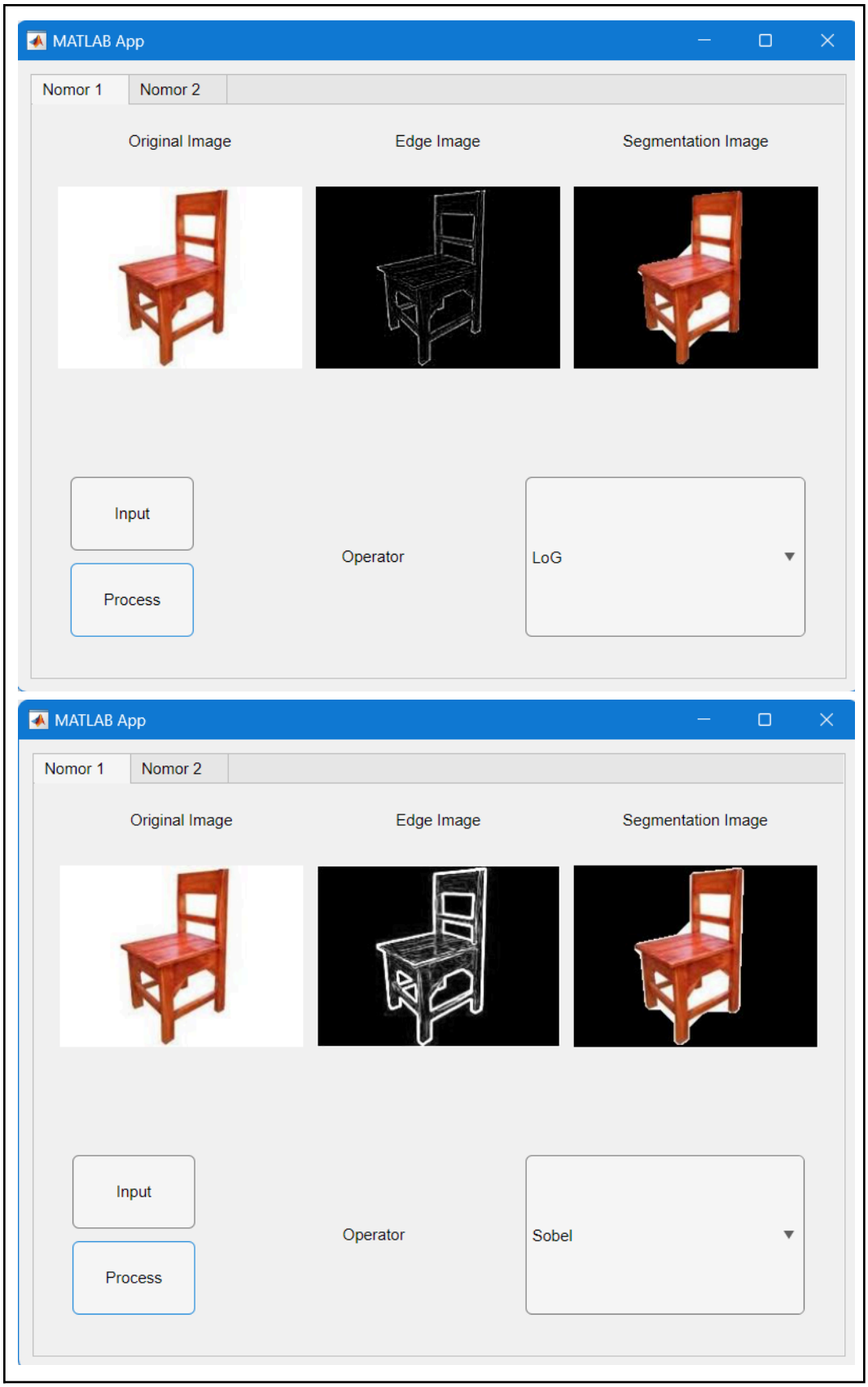


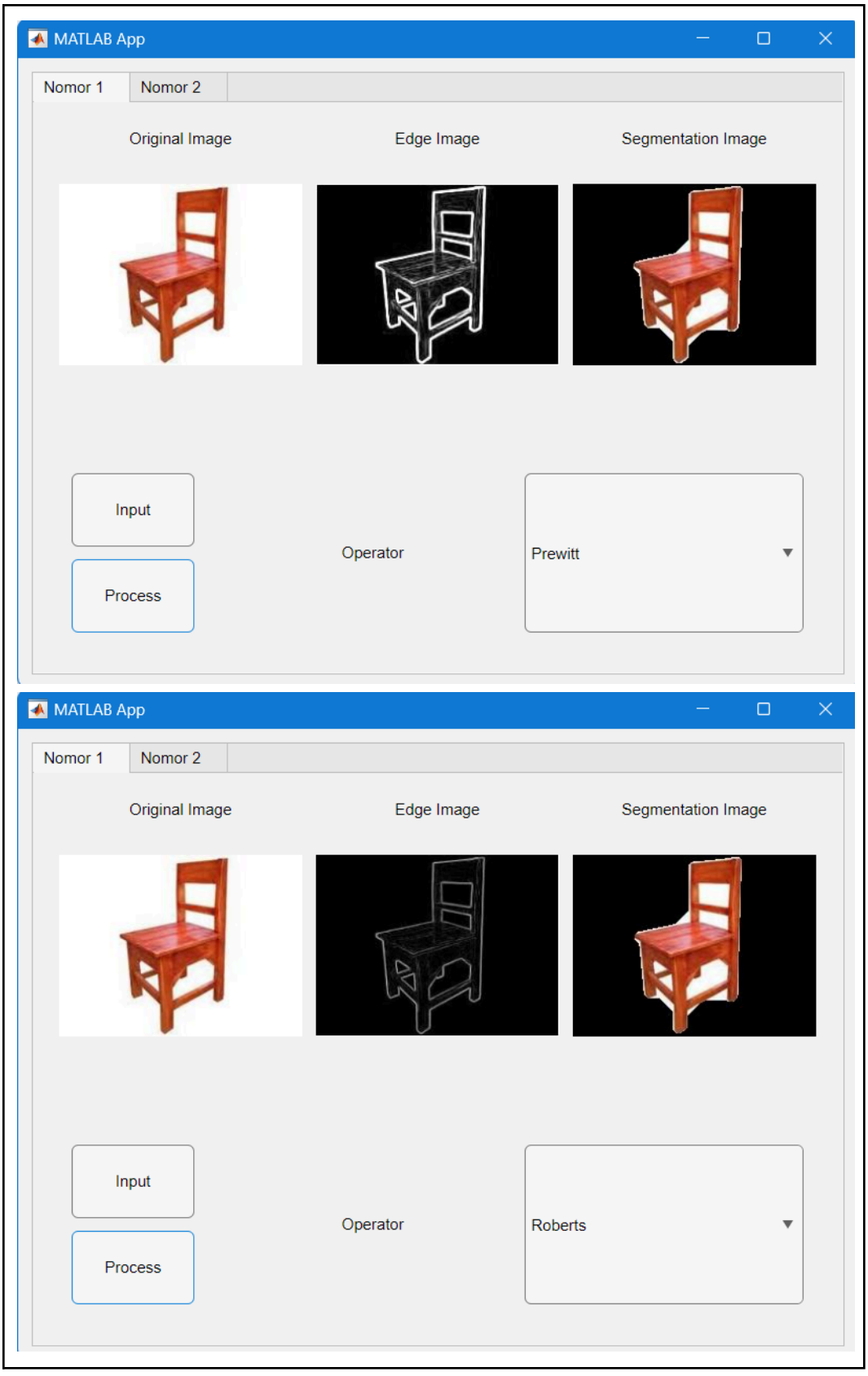

MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Sobel
MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Prewitt

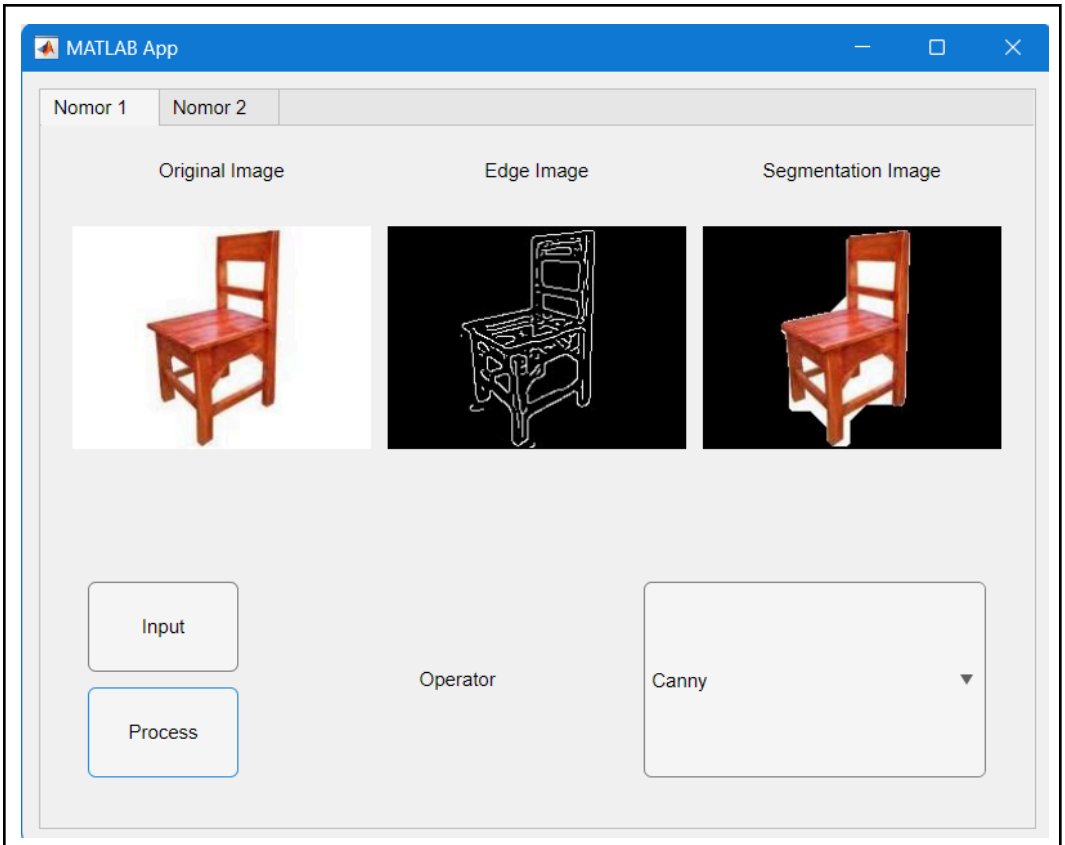

MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Roberts
MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Canny

chair.jpg

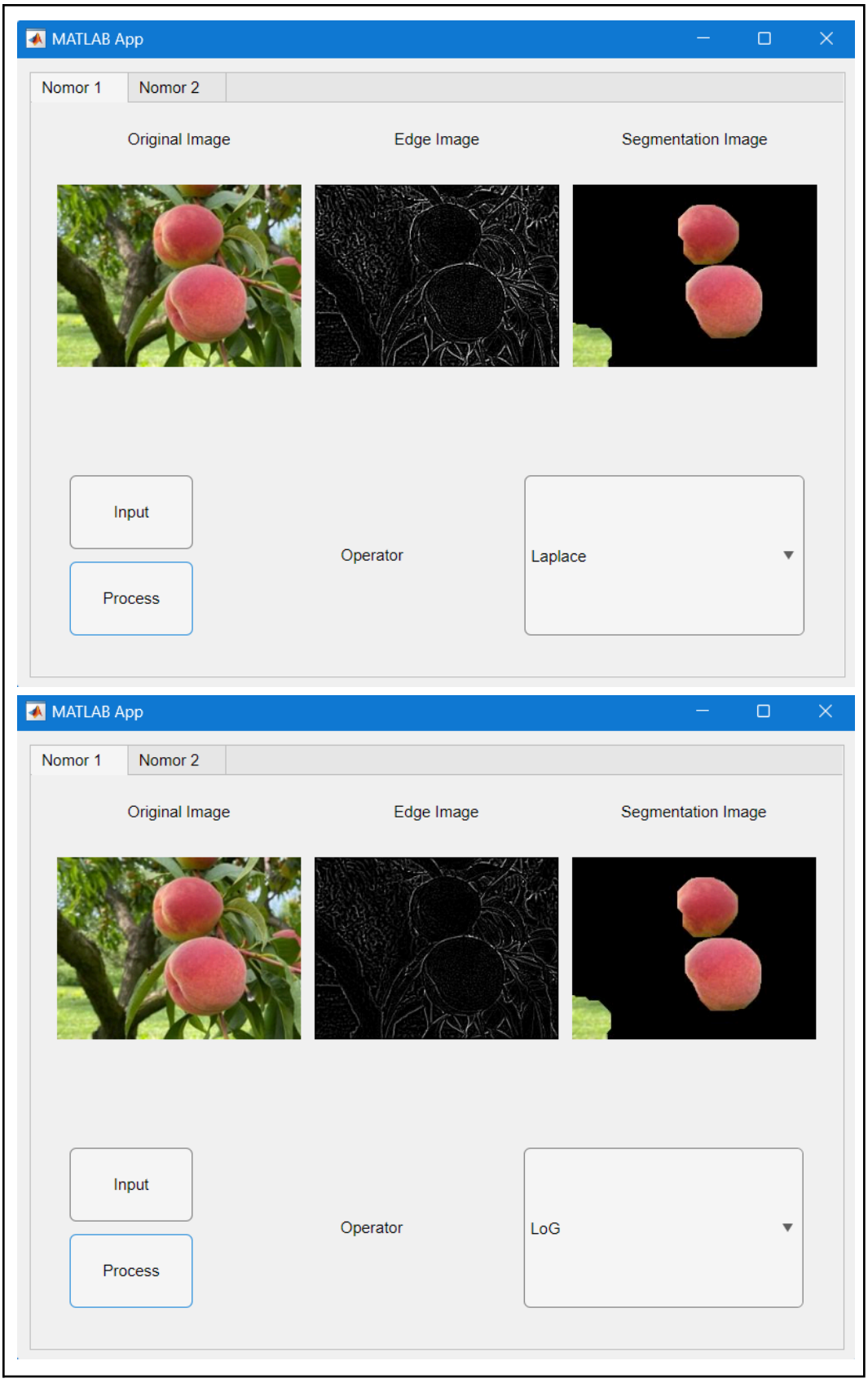

MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Laplace

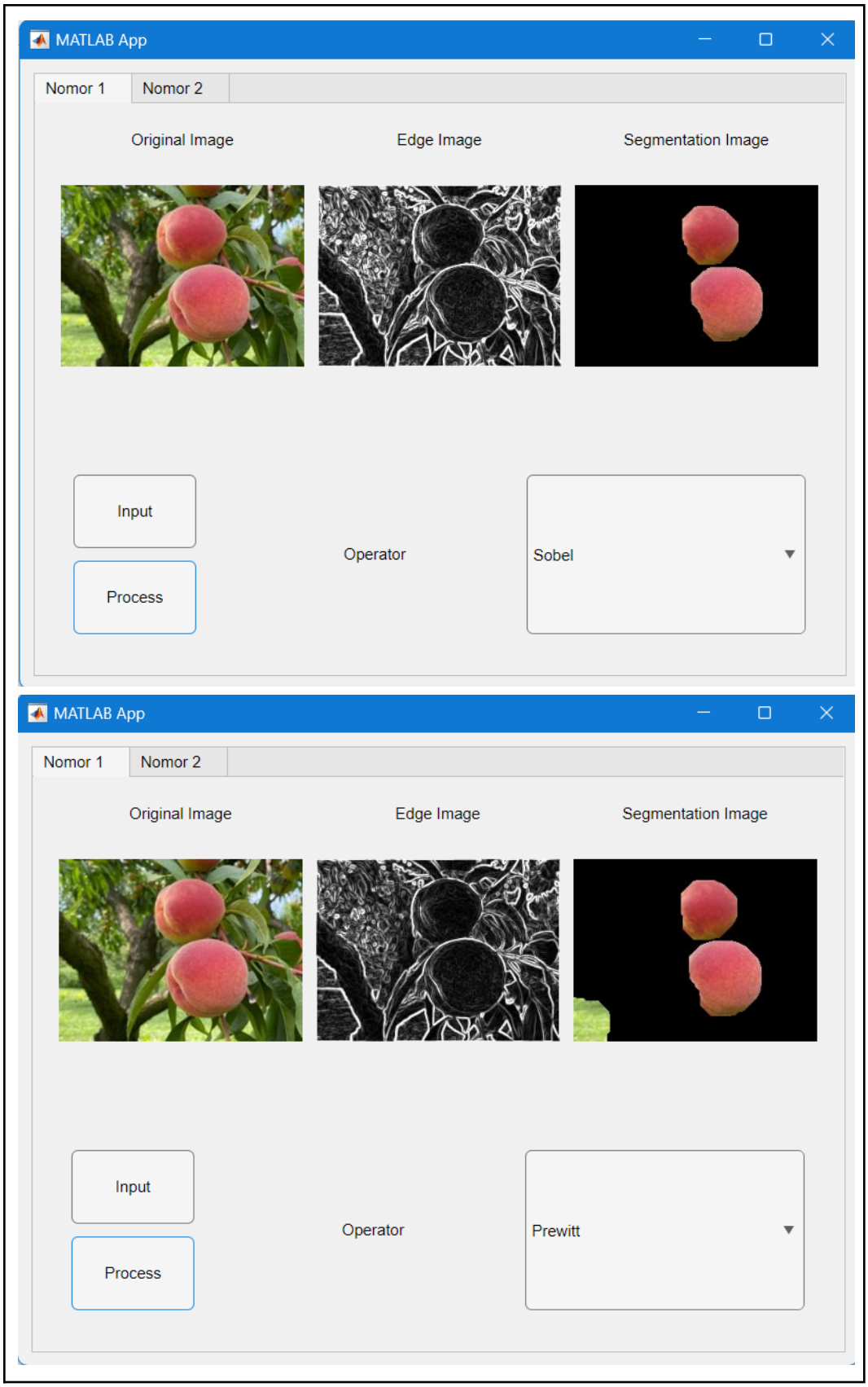

MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
LoG
MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Sobel

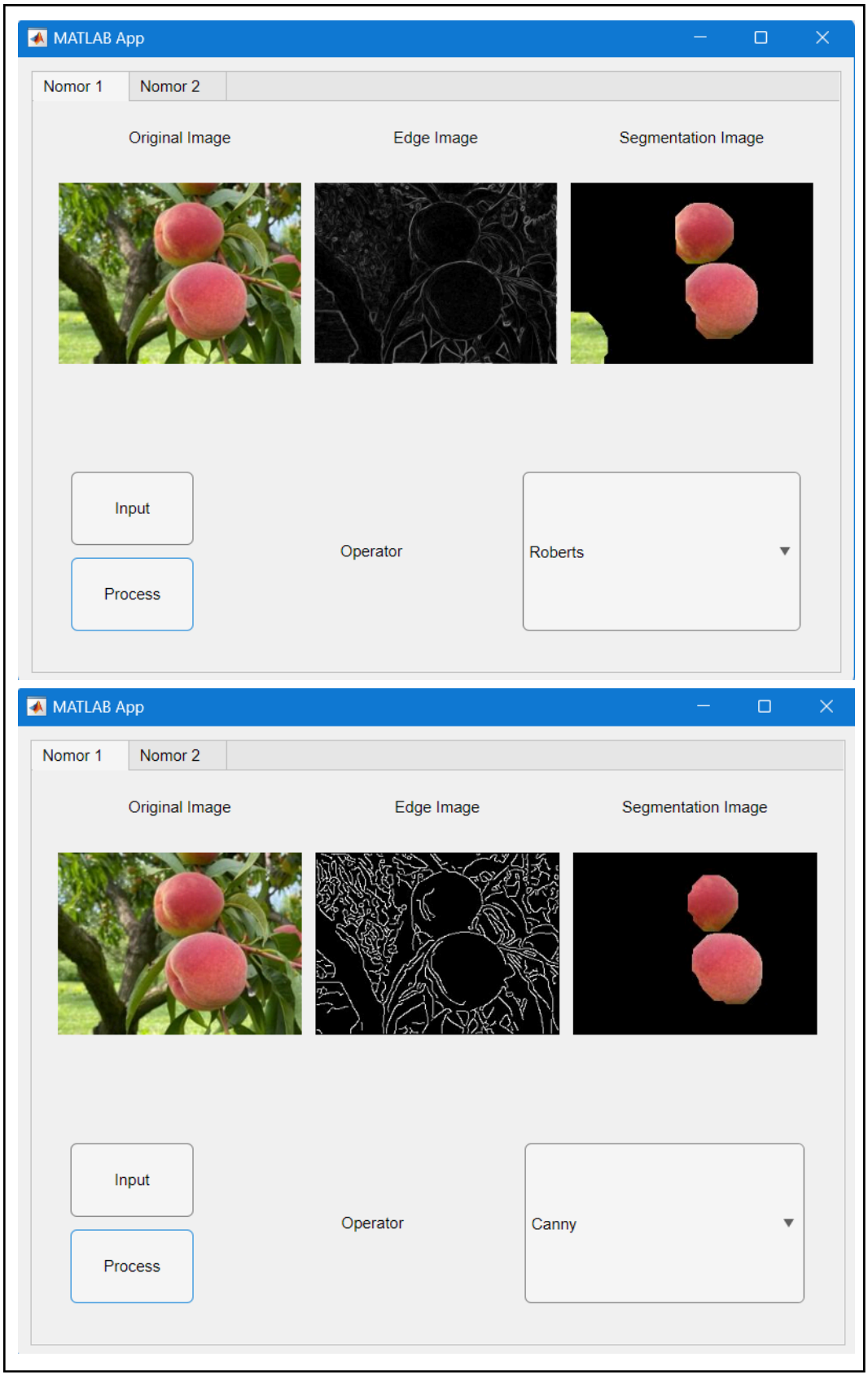

MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Prewitt
MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Roberts


MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Canny

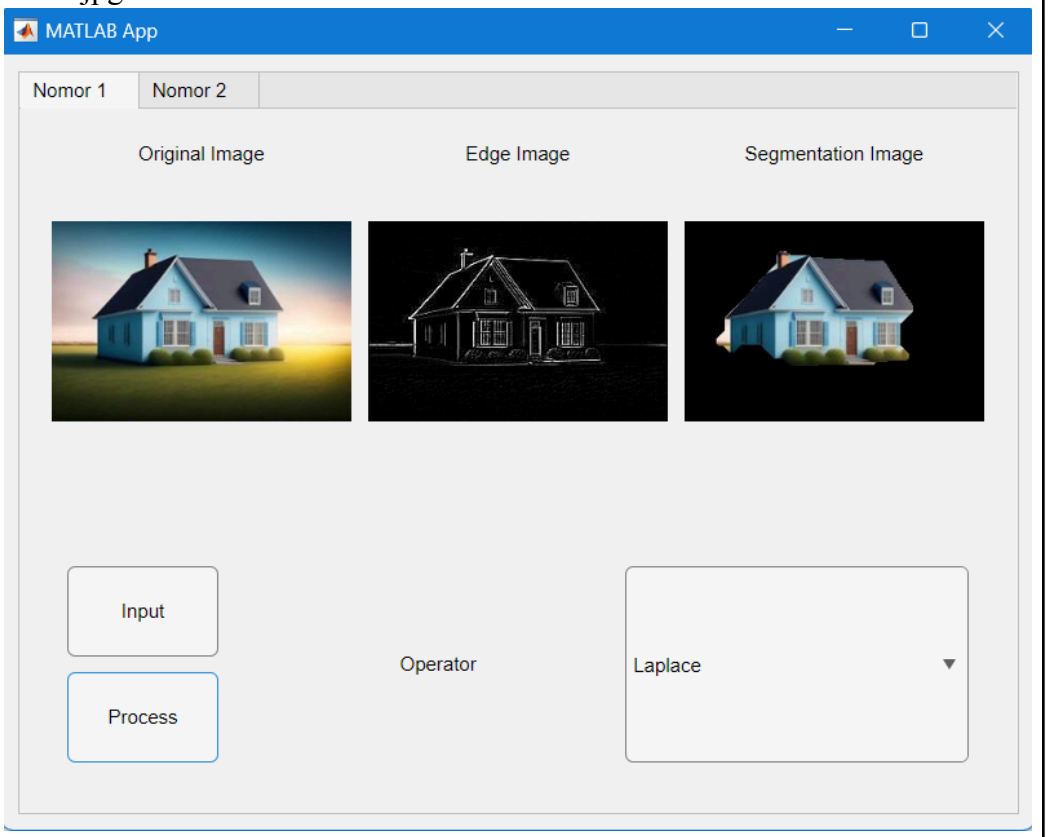
fruit.jpg

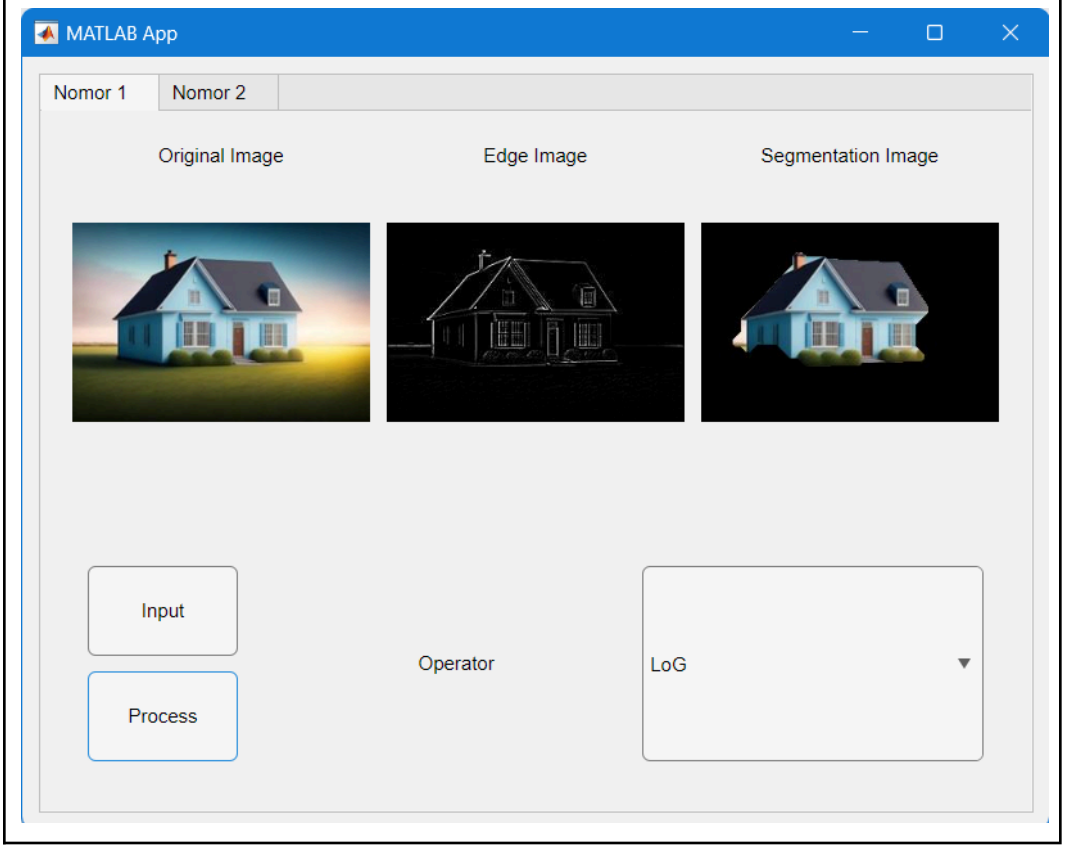

MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Laplace
MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
LoG

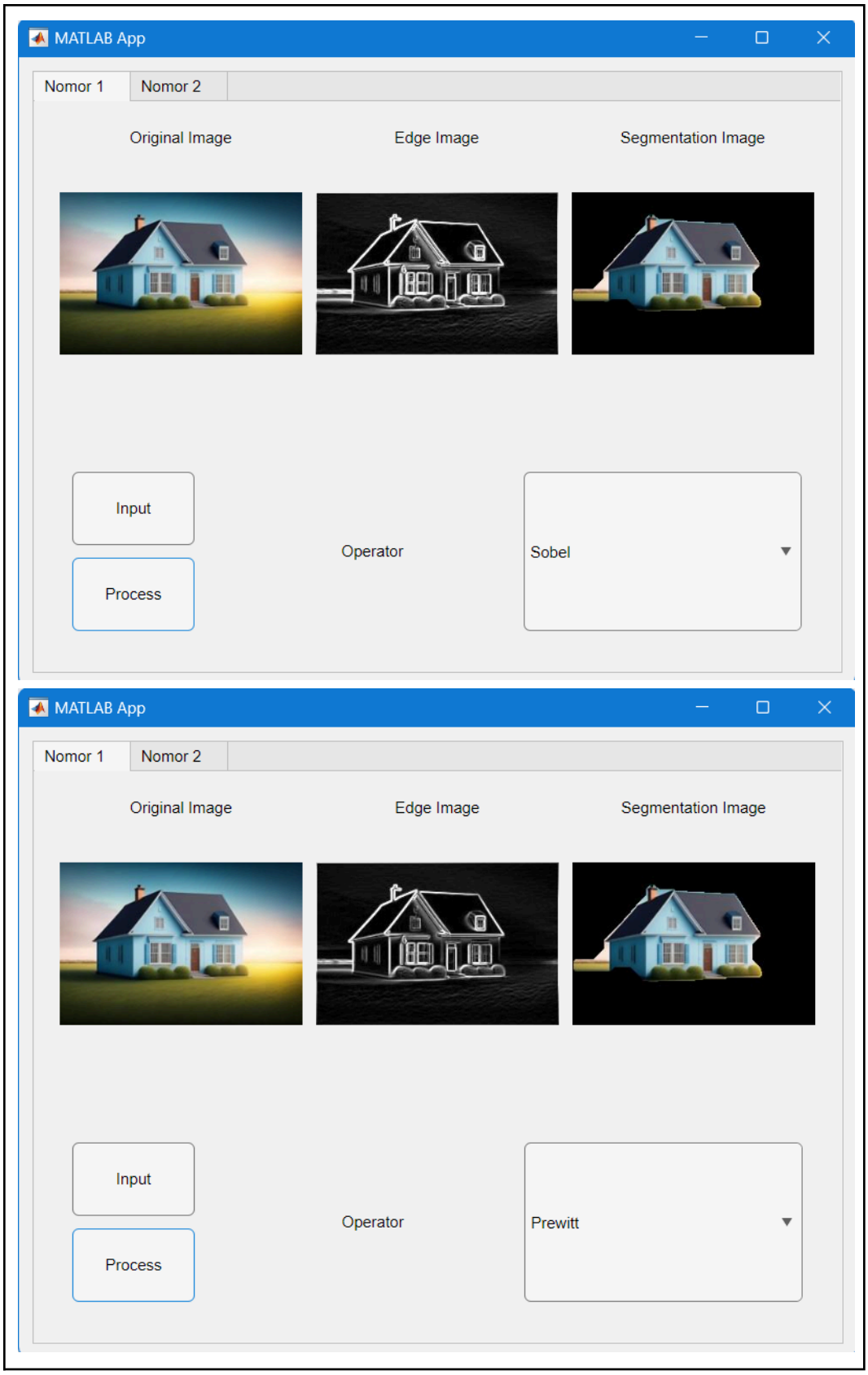

MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Sobel
MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Prewitt


MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Roberts
MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Canny

house.jpg


MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Laplace


MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
LoG


MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Sobel
MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Prewitt

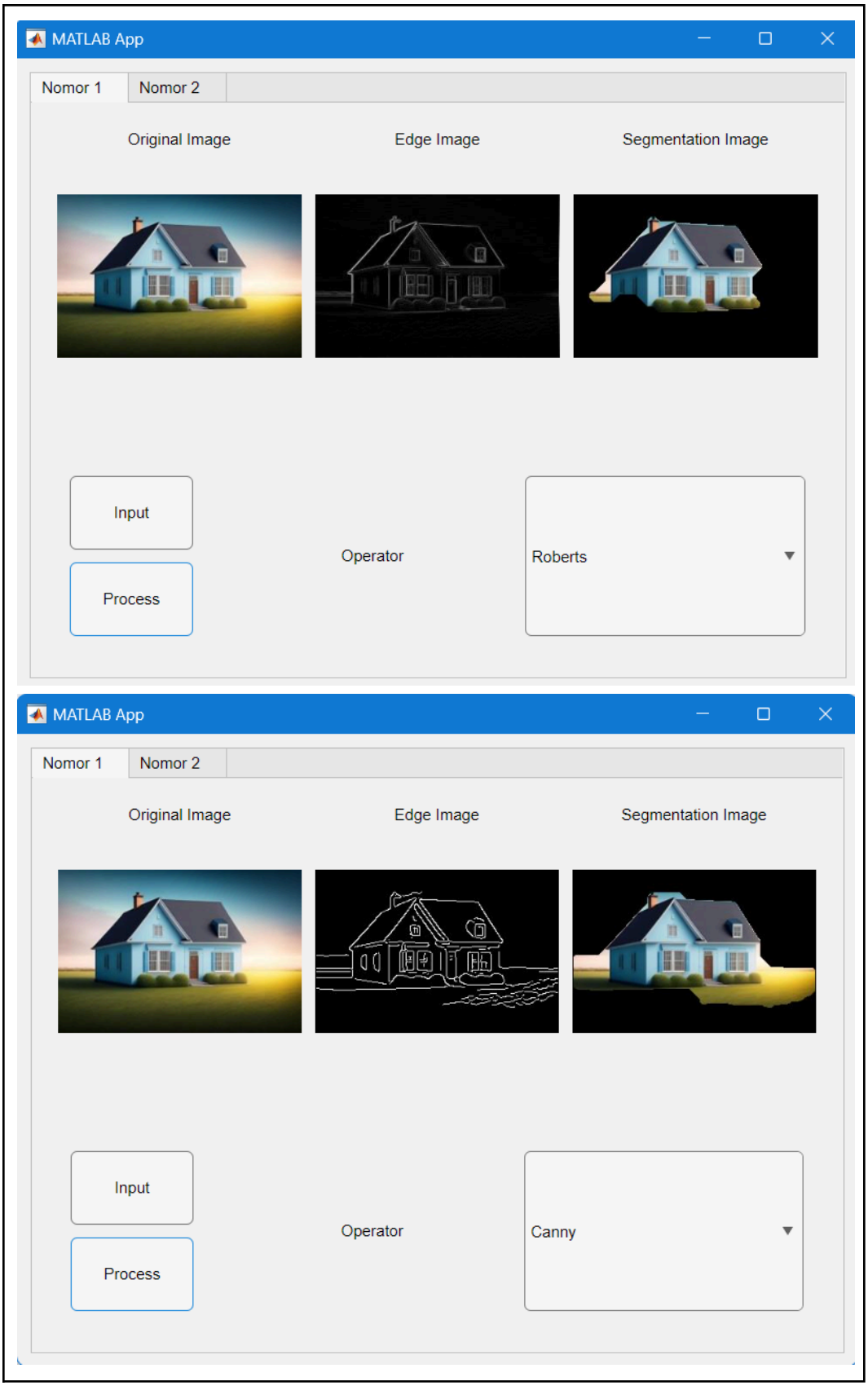
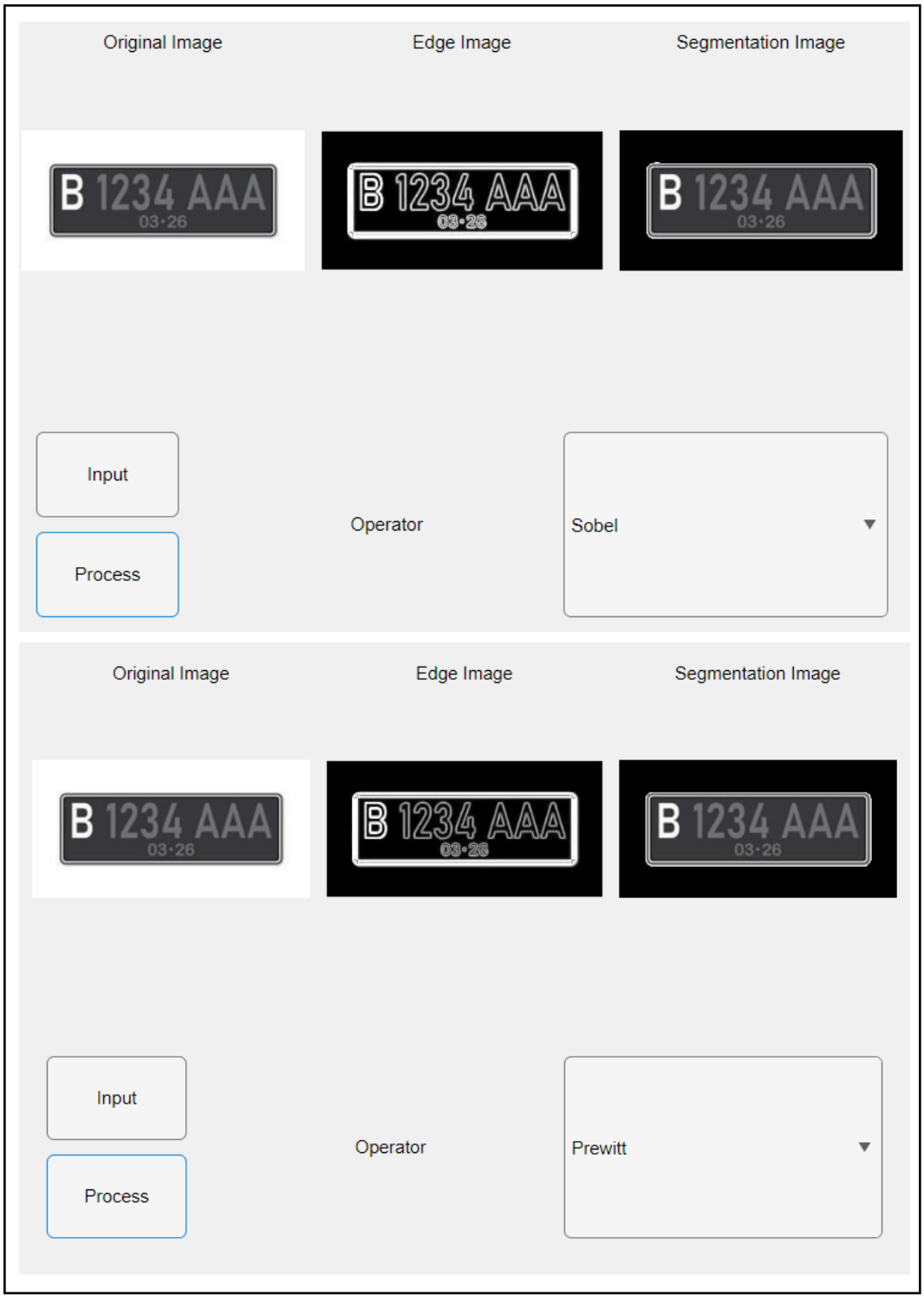
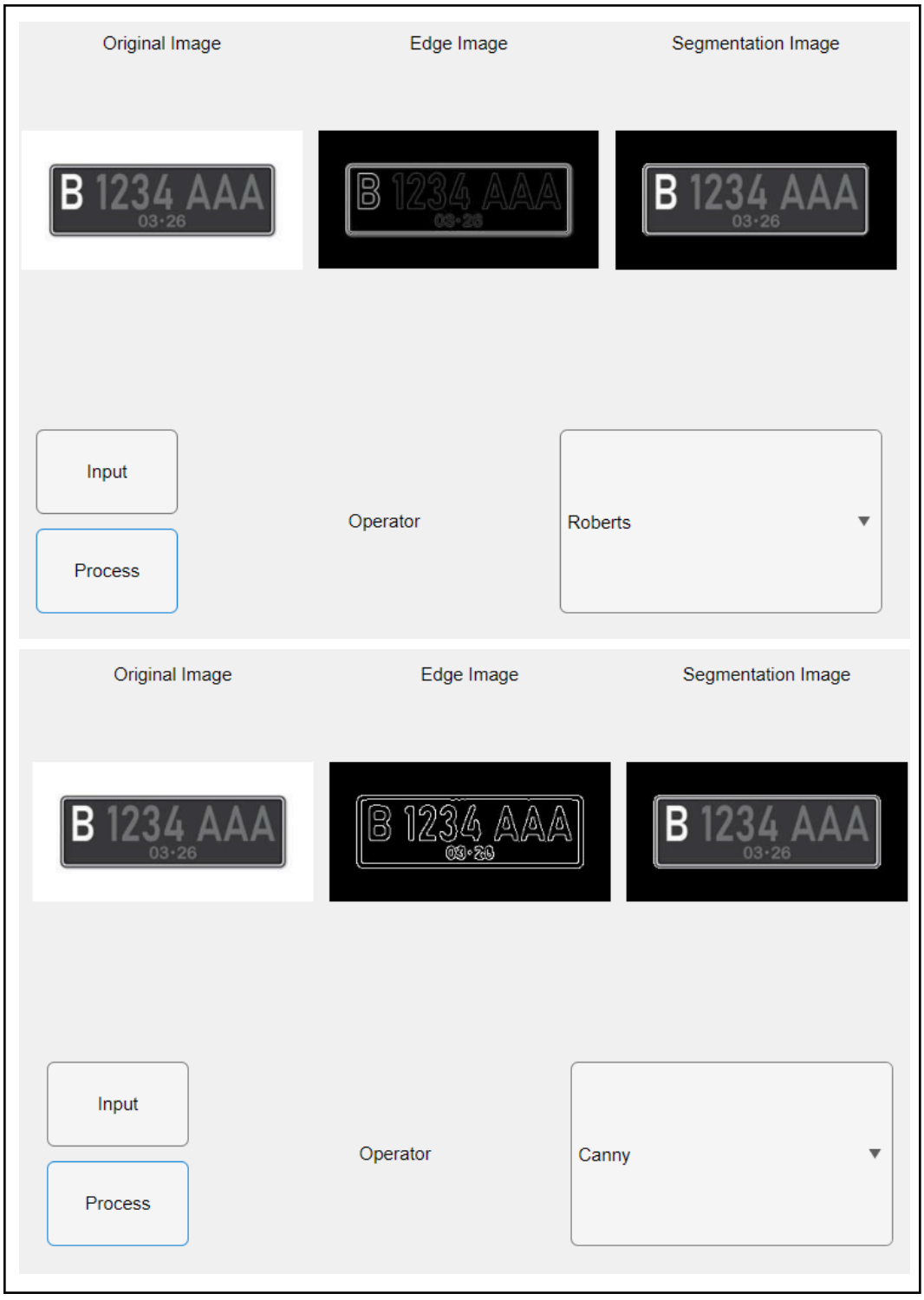

MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Roberts
MATLAB App
Nomor 1
Nomor 2
Original Image
Edge Image
Segmentation Image
Input
Process
Operator
Canny

plate.jpg


Original Image
Edge Image
Segmentation Image
B 1234 AAA
03·26
Input
Process
Operator
Laplace
Original Image
Edge Image
Segmentation Image
B 1234 AAA
03·26
Input
Process
Operator
LoG


Original Image
Edge Image
Segmentation Image
B 1234 AAA
03·26
Input
Process
Operator
Sobel
Original Image
Edge Image
Segmentation Image
B 1234 AAA
03·26
Input
Process
Operator
Prewitt



2.3. Analisis

Program untuk melakukan segmentasi objek dengan deteksi tepi dimulai dengan memperoleh gambar *edge* dengan operator yang dipilih. Terdapat 6 operator yang digunakan, yaitu Laplace, LoG, Sobel, Prewitt, Roberts, dan Canny. *Edge* yang

diperoleh masing-masing operator cukup beragam, ada yang menampilkan *edge* dengan sangat jelas seperti Canny, dan ada juga yang pada beberapa bagian kurang jelas seperti Laplace. Setelah diperoleh gambar *edge*, kemudian masuk ke proses segmentasi objek pada gambar. Segmentasi dimulai dengan mengubah border pada gambar *edge* menjadi hitam agar ketika dilakukan *filling*, gambar tidak menjadi putih semua. Lalu gambar *edge* yang grayscale diubah ke menjadi biner agar memudahkan proses segmentasi objek dari gambar aslinya. Setelah menjadi biner, dilakukan penghapusan pada komponen kecil di gambar *edge* agar segmentasi dapat difokuskan pada objek utama. *Edge* objek yang belum terhubung kemudian disambungkan supaya ketika dilakukan *filling*, seluruh bagian objek menjadi putih. Dilakukan kembali penghapusan pada komponen kecil untuk memastikan hanya objek utama yang muncul. Perlu diketahui bahwa semua nilai yang digunakan sebagai parameter pada proses ini diperoleh dari *trial and error*. Sekarang, gambar *edge* harusnya sudah berubah menjadi gambar segmentasi objek, dimana pada bagian objek berwarna putih penuh. Pada langkah terakhir, objek dipisahkan dari latar belakangnya dengan gambar segmentasi objek yang telah dibuat.

Dari test yang dilakukan, hasilnya sangat bervariasi. Beberapa gambar dapat disegmentasi dengan baik oleh semua operator, namun beberapa juga memberikan hasil yang buruk. Jika diperhatikan dari hasil segmentasi, gambar *edge* yang bagus dan yang tidak memiliki kelemahan masing-masing. Untuk gambar *edge* yang bagus, seringkali *edge* selain objek utama ikut tercipta, akibatnya hasil segmentasi tidak fokus pada objek dan mengikutsertakan bagian di sekitar objek. Sedangkan untuk *edge* yang kurang jelas, terkadang menyebabkan objek yang ingin disegmentasi terpotong. Selain itu, masalah yang belum bisa diatasi adalah terhadap objek yang bolong, seperti pada gambar chair.jpg. Hasil segmentasi tidak hanya memperoleh objek kursi saja, namun pada bagian yang bolong terisi gambar latar belakang. Hal ini disebabkan karena proses *filling* yang mengisi berdasarkan *edge* paling luar saja. Selain hal tersebut, program sudah memberikan performa yang baik dalam melakukan tugasnya.

3. Hough Transform Road Line Detection

3.1. Kode Program

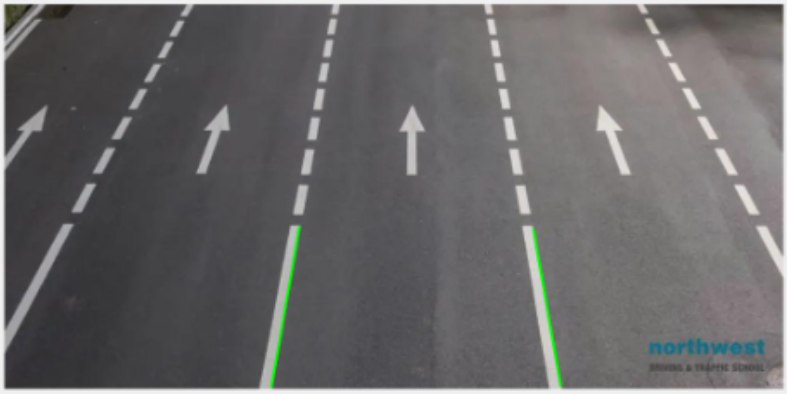
```
function outputImage = lineDetection(frame)
   %------------------Reading each frame from Video
File------------------
   % figure('Name','Original Image'), imshow(frame);
   frame = imgaussfilt3(frame);
   % figure('Name','Filtered Image'), imshow(frame);
```

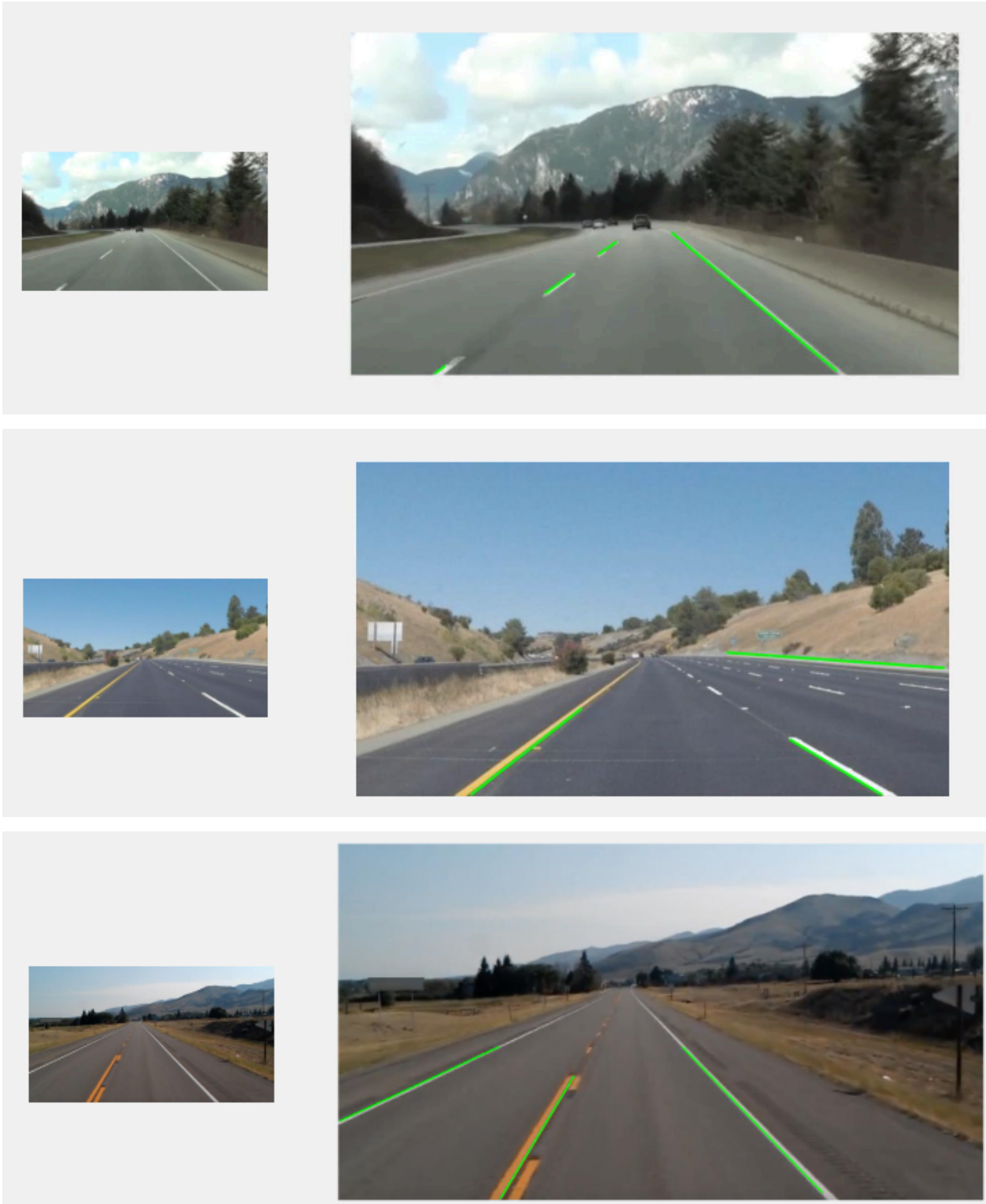
```
   % Convert the image to HSV color space
   hsvFrame = rgb2hsv(frame);

   % -------------- Define Thresholds for Masking Yellow Color
--------------

   % ---------------------- Define thresholds for 'Hue'
---------------------
   % Yellow hue is typically around 50 to 60 degrees in HSV color
space
   channel1MinY = 0.05;  % Corresponds to ~50° in hue (normalized
from 0 to 1)
   channel1MaxY = 0.23;  % Corresponds to ~60° in hue

   % ------------------ Define thresholds for 'Saturation'
------------------
   channel2MinY = 0.4;   % Minimum saturation for yellow
   channel2MaxY = 1;     % Maximum saturation for yellow

   % --------------------- Define thresholds for 'Value'
---------------------
   channel3MinY = 0.5;   % Minimum brightness for yellow
   channel3MaxY = 1;     % Maximum brightness for yellow

   % ----------- Create mask based on chosen histogram thresholds
-----------
   Yellow = (hsvFrame(:,:,1) >= channel1MinY) & (hsvFrame(:,:,1)
<= channel1MaxY) & ...
            (hsvFrame(:,:,2) >= channel2MinY) & (hsvFrame(:,:,2)
<= channel2MaxY) & ...
            (hsvFrame(:,:,3) >= channel3MinY) & (hsvFrame(:,:,3)
<= channel3MaxY);

   %figure('Name','Yellow Mask'), imshow(Yellow);

   % -------------- Define Thresholds for Masking White Color
--------------

   % ---------------------- Define thresholds for 'Hue'
---------------------
   % Hue for white doesn't matter, so you can set it to any
value.
   channel1MinW = 0;     % White can have any hue (from 0 to 1)
   channel1MaxW = 1;     % White can have any hue (from 0 to 1)

   % ------------------ Define thresholds for 'Saturation'
------------------
   channel2MinW = 0;     % White has low saturation (close to 0)
   channel2MaxW = 0.15;   % Saturation close to 0 (white is
desaturated)

   % --------------------- Define thresholds for 'Value'
---------------------
   channel3MinW = 0.7;   % White has high brightness (value close
```

```
to 1)
    channel3MaxW = 1;     % Maximum brightness for white

    % ----------- Create mask based on chosen histogram thresholds
-----------
    White = (hsvFrame(:,:,1) >= channel1MinW) & (hsvFrame(:,:,1)
<= channel1MaxW) & ...
            (hsvFrame(:,:,2) >= channel2MinW) & (hsvFrame(:,:,2)
<= channel2MaxW) & ...
            (hsvFrame(:,:,3) >= channel3MinW) & (hsvFrame(:,:,3)
<= channel3MaxW);

    %figure('Name','White Mask'), imshow(White);


    %imshow(White | Yellow)

    E = edge(White | Yellow, 'canny', 0.2);  % Edge detection with
adjusted thresholds
    % Get the size of the image
    [rows, cols] = size(E);

    % Create a mask to ignore the upper half of the image
    mask = false(rows, cols);
    mask(floor(rows*4/7):end, :) = true;  % Lower half is set to
true

    % Apply the mask to keep only the lower half
    E = E & mask;
    %figure('Name','Edge'), imshow(E)
    [H, T, R] = hough(E);  % Hough Transform
    % Calculate NHoodSize as half of the Hough matrix dimensions,
ensuring it's odd
    [rows, cols] = size(H);
    nhoodRows = 2 * floor(rows / 10) + 1; % Half of rows (rounded
down and made odd)
    nhoodCols = 2 * floor(cols / 10) + 1; % Half of cols (rounded
down and made odd)
    nhoodSize = [nhoodRows, nhoodCols];

    % Find Hough Peaks
    P = houghpeaks(H, 3, "Threshold", ceil(0.3 * max(H(:))),
"NHoodSize", nhoodSize);  % Detect peaks

    % Hough lines detection
    lines = houghlines(E, T, R, P, 'FillGap', 30, 'MinLength',
20);

    % Display results
    fig = figure('visible', 'off');
    imshow(frame), hold on;
    for k = 1:length(lines)
        xy = [lines(k).point1; lines(k).point2];  % Corrected
field names
        plot(xy(:,1), xy(:,2), 'LineWidth', 2, 'Color', 'green');
% Adjust line width
```

```
    end

    frameCapture = getframe(fig);
    outputImage = frameCapture.cdata;

    % Close the hidden figure
    close(fig);
end
```

3.2. Contoh Hasil Eksekusi





3.3. Analisis

Transformasi hough mendeteksi garis pada suatu image dengan mencari titik yang akan beririsan dengan sinusoid pada parameter space dengan fase dan fungsi sinusoid yang sama. Untuk mendeteksi garis jalan menggunakan transformasi ini,

tepi gambar perlu dicari untuk mendapatkan garis yang lebih jelas. Namun, tepi saja ternyata tidak cukup untuk mendeteksi garis jalan dengan akurat karena pengaruh pemandangan pada gambar dan hal lain yang menjadi salah satu puncak pada hasil transformasi hough, sehingga image yang akan digunakan akan melewati suatu tahap *preprocessing*. Pada tugas ini, kami melakukan suatu masking terhadap warna kuning dan putih pada HSV space dengan asumsi variasi warna garis jalan hanya putih dan kuning. Meskipun dirasa cukup, ternyata hal ini juga belum memuaskan dikarenakan pemandangan yang juga dapat berwarna putih atau kuning, sehingga dilakukan juga *cut off* pada 4/7 bagian teratas gambar sehingga pemandangan yang akan menghasilkan tepi lurus dapat dihilangkan. Setelah melakukan semua hal tersebut, ternyata masih belum cukup bagus, masih banyak garis hough *peaks* yang terdeteksi berulang pada daerah garis yang sama, oleh karena itu diberikan parameter NHoodSpace untuk tidak mengikut sertakan *peaks* pada rentang daerah tertentu. Beberapa garis juga memiliki hasil tepi yang putus-putus sehingga digunakan juga FillGap untuk menyambungkannya. Dengan menggunakan min length, garis-garis *noise* juga dapat dihilangkan.

Lampiran

Link github: https://github.com/ChrisAlberth/Tugas3_PCD

MATLAB GUI - MATLAB & Simulink (mathworks.com)