

Tugas 3 IF4073 Pemrosesan Citra Digital
Segmentasi Objek, Deteksi Tepi, dan Transformasi Hough



Disusun Oleh:

13521059 Arleen Chrysantha Gunardi

13521107 Jericho Russel Sebastian

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2024

Daftar Isi

Daftar Isi.....	1
Daftar Gambar.....	2
1. GUI.....	4
2. Program.....	4
2.1. Segmentasi Objek dengan Deteksi Tepi.....	4
2.2. Deteksi Jalur Garis di Jalan Raya dengan Transformasi Hough.....	35
3. Lampiran.....	42

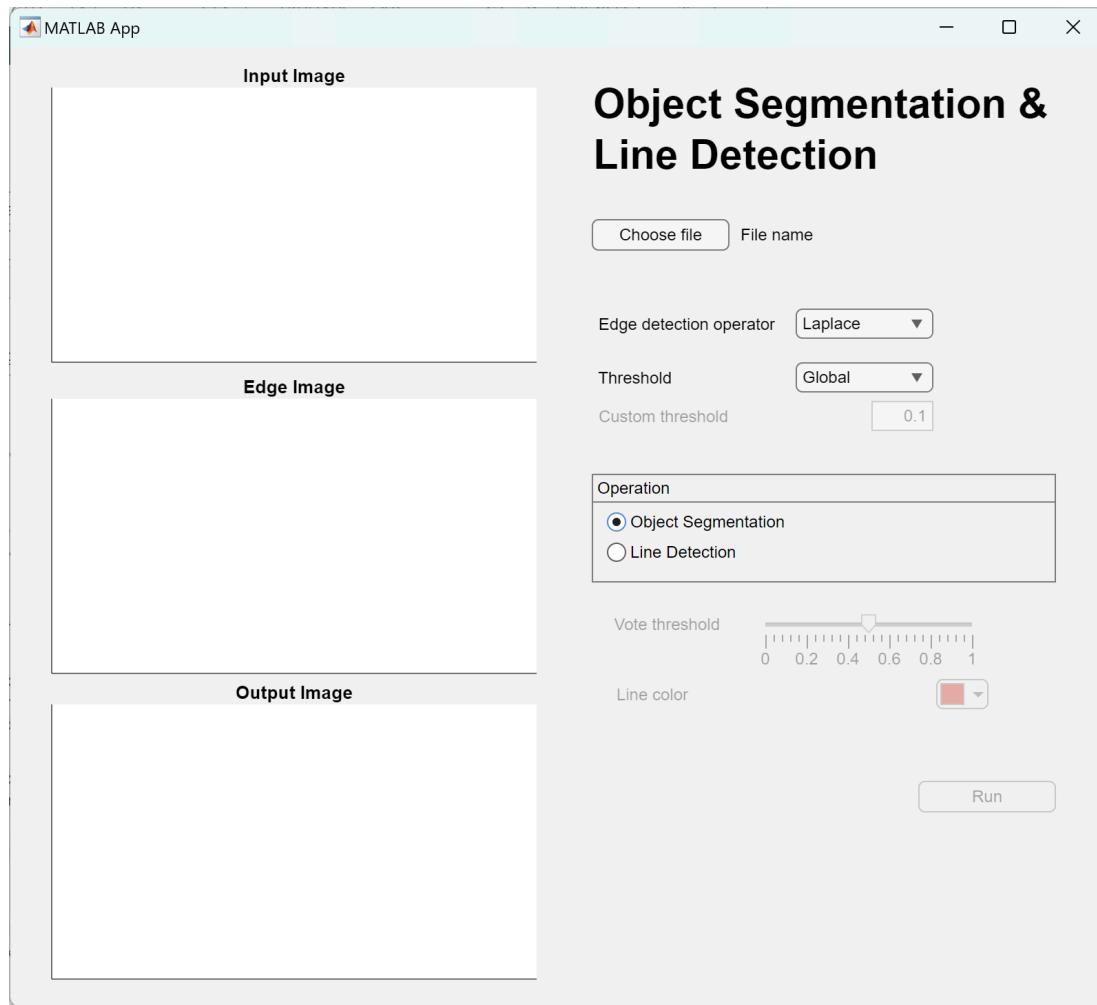
Daftar Gambar

Gambar 1.1 GUI Program.....	4
Gambar 2.1 Segmentasi objek citra mobil dengan operator Laplace.....	8
Gambar 2.2 Segmentasi objek citra mobil dengan operator Laplace of Gaussian.....	9
Gambar 2.3 Segmentasi objek citra mobil dengan operator Sobel.....	9
Gambar 2.4 Segmentasi objek citra mobil dengan operator Prewitt.....	10
Gambar 2.5 Segmentasi objek citra mobil dengan operator Roberts.....	10
Gambar 2.6 Segmentasi objek citra mobil dengan operator Canny.....	11
Gambar 2.7 Segmentasi objek citra buah persik dengan operator Laplace.....	11
Gambar 2.8 Segmentasi objek citra buah persik dengan operator Laplace of Gaussian.....	12
Gambar 2.9 Segmentasi objek citra buah persik dengan operator Sobel.....	12
Gambar 2.10 Segmentasi objek citra buah persik dengan operator Prewitt.....	13
Gambar 2.11 Segmentasi objek citra buah persik dengan operator Roberts.....	13
Gambar 2.12 Segmentasi objek citra buah persik dengan operator Canny.....	14
Gambar 2.13 Segmentasi objek citra rumah dengan operator Laplace.....	14
Gambar 2.14 Segmentasi objek citra rumah dengan operator Laplace of Gaussian.....	15
Gambar 2.15 Segmentasi objek citra rumah dengan operator Sobel.....	15
Gambar 2.16 Segmentasi objek citra rumah dengan operator Prewitt.....	16
Gambar 2.17 Segmentasi objek citra rumah dengan operator Roberts.....	16
Gambar 2.18 Segmentasi objek citra rumah dengan operator Canny.....	17
Gambar 2.19 Segmentasi objek citra alpukat dengan operator Laplace.....	17
Gambar 2.20 Segmentasi objek citra alpukat dengan operator Laplace of Gaussian.....	18
Gambar 2.21 Segmentasi objek citra alpukat dengan operator Sobel.....	18
Gambar 2.22 Segmentasi objek citra alpukat dengan operator Prewitt.....	19
Gambar 2.23 Segmentasi objek citra alpukat dengan operator Roberts.....	19
Gambar 2.24 Segmentasi objek citra alpukat dengan operator Canny.....	20
Gambar 2.25 Segmentasi objek citra pelat nomor kendaraan dengan operator Laplace.....	20
Gambar 2.26 Segmentasi objek citra pelat nomor kendaraan dengan operator Laplace of Gaussian.....	21
Gambar 2.27 Segmentasi objek citra pelat nomor kendaraan dengan operator Sobel.....	21
Gambar 2.28 Segmentasi objek citra pelat nomor kendaraan dengan operator Prewitt.....	22
Gambar 2.29 Segmentasi objek citra pelat nomor kendaraan dengan operator Roberts.....	22
Gambar 2.30 Segmentasi objek citra pelat nomor kendaraan dengan operator Canny.....	23
Gambar 2.31 Segmentasi objek citra pisang dengan operator Laplace.....	23
Gambar 2.32 Segmentasi objek citra pisang dengan operator Laplace of Gaussian.....	24
Gambar 2.33 Segmentasi objek citra pisang dengan operator Sobel.....	24
Gambar 2.34 Segmentasi objek citra pisang dengan operator Prewitt.....	25

Gambar 2.35 Segmentasi objek citra pisang dengan operator Roberts.....	25
Gambar 2.36 Segmentasi objek citra pisang dengan operator Canny.....	26
Gambar 2.37 Segmentasi objek citra mobil dan motor dengan operator Laplace.....	26
Gambar 2.38 Segmentasi objek citra mobil dan motor dengan operator Laplace of Gaussian.....	27
Gambar 2.39 Segmentasi objek citra mobil dan motor dengan operator Sobel.....	27
Gambar 2.40 Segmentasi objek citra mobil dan motor dengan operator Prewitt.....	28
Gambar 2.41 Segmentasi objek citra mobil dan motor dengan operator Roberts.....	28
Gambar 2.42 Segmentasi objek citra mobil dan motor dengan operator Canny.....	29
Gambar 2.43 Segmentasi objek citra kursi dengan operator Laplace.....	29
Gambar 2.44 Segmentasi objek citra kursi dengan operator Laplace of Gaussian.....	30
Gambar 2.45 Segmentasi objek citra kursi dengan operator Sobel.....	30
Gambar 2.46 Segmentasi objek citra kursi dengan operator Prewitt.....	31
Gambar 2.47 Segmentasi objek citra kursi dengan operator Roberts.....	31
Gambar 2.48 Segmentasi objek citra kursi dengan operator Canny.....	32
Gambar 2.49 Segmentasi objek citra kucing dengan operator Prewitt.....	32
Gambar 2.50 Segmentasi objek citra burung dengan operator Canny.....	33
Gambar 2.51 Deteksi jalur garis jalan raya 1.....	38
Gambar 2.52 Deteksi jalur garis jalan raya 2.....	38
Gambar 2.53 Deteksi jalur garis jalan raya 3.....	39
Gambar 2.54 Deteksi jalur garis jalan raya 4.....	39
Gambar 2.55 Deteksi jalur garis jalan raya 5 dengan operator Canny.....	40
Gambar 2.56 Deteksi jalur garis jalan raya 5 dengan operator Sobel.....	40
Gambar 2.57 Deteksi jalur garis jalan raya 6.....	41

1. GUI

Berikut adalah tangkapan layar untuk tampilan GUI.



Gambar 1.1 GUI Program

2. Program

2.1. Segmentasi Objek dengan Deteksi Tepi

Berikut merupakan kode program fungsi untuk deteksi tepi terhadap citra.

```
function result = edgeDetection(image, operator, thresholdType,  
threshold)  
    if size(image, 3) == 3  
        image = rgb2gray(image);  
    end  
  
    switch lower(operator)  
        case 'laplace'  
            %kernelX = [0 1 0; 1 -4 1; 0 1 0];  
            kernelX = [1 1 1; 1 -8 1; 1 1 1];
```

```

%kernelX = [0 -1 0; -1 4 -1; 0 -1 0];
%kernelX = [-1 -1 -1; -1 8 -1; -1 -1 -1];
edges = conv2(double(image), double(kernelX), 'valid');
case 'log'
    kernelX = [0 0 -1 0 0;
               0 -1 -2 -1 0;
               -1 -2 16 -2 -1;
               0 -1 -2 -1 0;
               0 0 -1 0 0];
    edges = conv2(double(image), double(kernelX), 'valid');
case 'sobel'
    kernelX = [-1 0 1; -2 0 2; -1 0 1];
    kernelY = [1 2 1; 0 0 0; -1 -2 -1];
    convX = conv2(double(image), double(kernelX), 'valid');
    convY = conv2(double(image), double(kernelY), 'valid');
    edges = sqrt(convX.^2 + convY.^2);
case 'prewitt'
    kernelX = [-1 0 1; -1 0 1; -1 0 1];
    kernelY = [-1 -1 -1; 0 0 0; 1 1 1];
    convX = conv2(double(image), double(kernelX), 'valid');
    convY = conv2(double(image), double(kernelY), 'valid');
    edges = sqrt(convX.^2 + convY.^2);
case 'roberts'
    kernelX = [1 0; 0 -1];
    kernelY = [0 1; -1 0];
    convX = conv2(double(image), double(kernelX), 'valid');
    convY = conv2(double(image), double(kernelY), 'valid');
    edges = sqrt(convX.^2 + convY.^2);
case 'canny'
    edges = (edge(image, 'canny') * 255);
otherwise
    error('Unknown operator: %s', operator);
end
if strcmpi(operator, 'canny')
    result = edges;
else
    % Add some padding
    [imageHeight, imageWidth] = size(image);
    [kernelXHeight, kernelXWidth] = size(kernelX);
    paddingHeight = floor(kernelXHeight / 2);
    paddingWidth = floor(kernelXWidth / 2);

    result = zeros(imageHeight, imageWidth);
    if strcmpi(operator, 'roberts')
        result(paddingHeight:imageHeight - paddingHeight,
paddingWidth:imageWidth - paddingWidth) = edges;
    else
        result(paddingHeight + 1:imageHeight - paddingHeight,
paddingWidth + 1:imageWidth - paddingWidth) = edges;
    end
    % thresholding the edge mask
    switch lower(thresholdType)
        case 'adaptive'
            result = imbinarize(result, 'adaptive');
        case 'otsu'
            result = imbinarize(result, graythresh(image));

```

```

        case 'global'
            result = imbinarize(result, 'global');
        case 'input'
            result = imbinarize(result, threshold);
        otherwise
            error('Unknown threshold type: %s', thresholdType);
    end
end

```

Fungsi `edgeDetection` menerima empat buah parameter, yaitu `image` (citra masukan), `operator` (jenis operator), `thresholdType` (jenis `threshold`), dan `threshold` (nilai `threshold` untuk jenis `threshold` input). Adapun parameter `operator` pada fungsi ini adalah jenis operator yang ingin digunakan, yaitu operator Laplace, Laplace of Gaussian (LoG), Sobel, Prewitt, Roberts, dan Canny. Sementara itu, parameter `thresholdType` merupakan jenis `threshold` untuk proses konversi citra tepi menjadi citra biner, yaitu *global*, *adaptive*, *Otsu*, dan `threshold` yang nilainya berasal dari masukan pengguna (parameter `threshold`).

Pertama-tama, citra (`image`) yang berwarna dibuat menjadi citra abu (*grayscale*) terlebih dahulu. Kemudian, citra dikonvolusikan dengan *kernel* sesuai dengan masing-masing operator yang dipilih (kecuali untuk operator Canny yang menggunakan fungsi bawaan dari Matlab). Operasi konvolusi dilakukan dengan fungsi bawaan Matlab, yaitu `conv2` dengan parameter `shape` bernilai *valid* agar konvolusi dilakukan tanpa menambahkan *padding* pada sisi citra. Untuk mendapatkan deteksi tepi yang lebih baik, hasil konvolusi ditambahkan *padding* dengan ukuran setengah dari ukuran *kernel*. Kemudian, hasil deteksi tepi tersebut dikonversikan menjadi citra biner dengan teknik pengambangan sesuai parameter fungsi (*global*, *adaptive*, *Otsu*, dan nilai `threshold` tertentu). Hasil deteksi tepi yang dikembalikan fungsi merupakan citra biner tepi dengan *padding*.

Citra biner yang berisi tepi objek pada citra akan digunakan untuk melakukan segmentasi objek pada citra. Berikut merupakan kode program fungsi untuk melakukan segmentasi objek.

```

function segmentedImage = segmentation(image, edges)
    mask = imclearborder(edges);
    mask = imclose(mask, strel('line', 10,0)); % close (connect)
adjacent edges
    mask = imfill(mask, 'holes'); % fill areas inside connected
edges
    mask = imopen(mask, strel(ones(3, 3)));
    mask = bwareaopen(mask, 1500); % remove filled areas under
threshold (background)
    if size(image, 3) == 3
        % RGB image
        segmentedImage(:,:,:,1) = image(:,:,:,1) .* mask;
        segmentedImage(:,:,:,2) = image(:,:,:,2) .* mask;
        segmentedImage(:,:,:,3) = image(:,:,:,3) .* mask;
    else
        % grayscale image
        segmentedImage(:,:,:) = image(:,:,:) .* mask;
    end
end

```

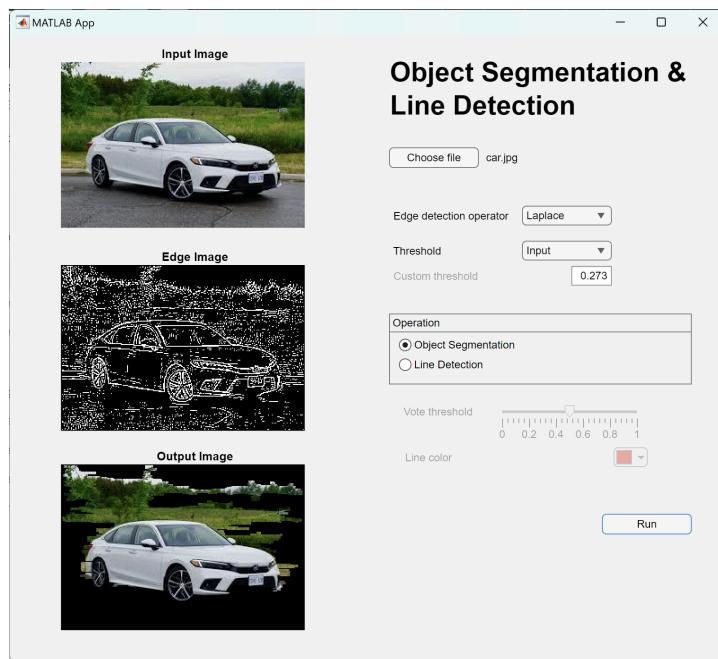
Fungsi `segmentation` menerima dua parameter, yaitu `image` (citra masukan) dan `edges` (citra biner tepi). Pertama-tama, citra tepi (`edges`) diolah menjadi `mask` dengan cara menghapuskan tepian (*border*) terlebih dahulu dengan fungsi bawaan Matlab `imclearborder`, lalu menghubungkan tepian yang berdekatan dengan fungsi `imclose`, sehingga tepian dapat menutup dan area di dalamnya dapat diisi. Setelah tepian terhubung, area tertutup di dalam tepian diisi menggunakan fungsi `imfill` untuk memastikan tidak ada lubang di dalam objek yang terdeteksi. Kemudian, `mask` diproses dengan operasi morfologi *open* menggunakan fungsi `imopen`, yang menghilangkan derau kecil dengan menggunakan elemen struktural berbentuk matriks 3×3 berisi nilai 1. Selanjutnya, fungsi `bwareaopen` digunakan untuk menghapus area-area kecil pada `mask` yang ukurannya kurang dari 1500 piksel. Tujuannya untuk menghilangkan bagian yang tidak relevan, seperti derau kecil atau objek yang terlalu kecil untuk dianggap signifikan.

Setelah `mask` selesai diproses, fungsi mengecek apakah citra masukan (`image`) berupa citra berwarna atau *grayscale*. Jika citra berwarna, `mask` diterapkan ke setiap kanal warna (merah, hijau, dan biru) secara terpisah. Sebaliknya, jika citra masukan *grayscale*, `mask` langsung dikalikan dengan citra masukan. Akhirnya, fungsi ini menghasilkan citra segmentasi yang hanya mempertahankan bagian objek yang

relevan (sesuai dengan *mask*) dan mengisi area sisanya dengan warna hitam (bernilai nol).

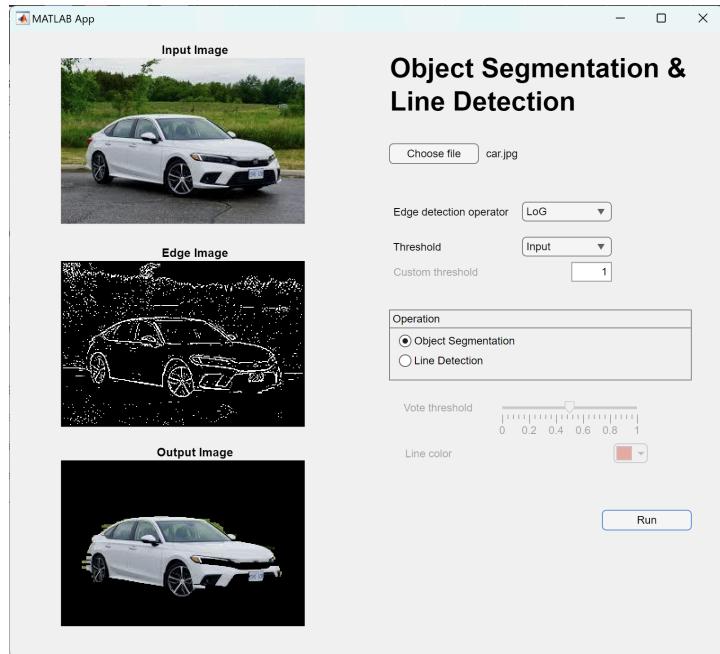
Berikut merupakan contoh hasil eksekusi program.

Segmentasi objek citra mobil (*car.jpg*) dengan operator *Laplace* dan *threshold* bernilai 0.273:



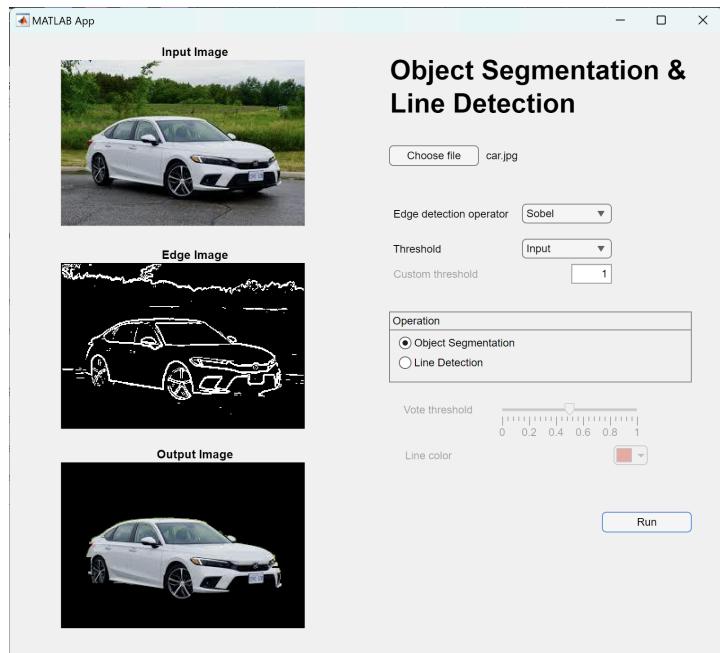
Gambar 2.1 Segmentasi objek citra mobil dengan operator *Laplace*

Segmentasi objek citra mobil (*car.jpg*) dengan operator *Laplace of Gaussian* dan *threshold* bernilai 1:



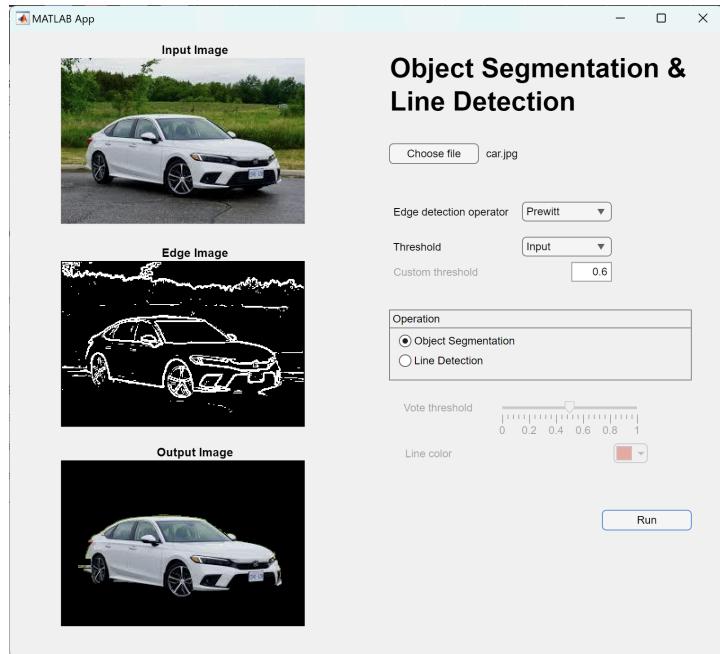
Gambar 2.2 Segmentasi objek citra mobil dengan operator *Laplace of Gaussian*

Segmentasi objek citra mobil (*car.jpg*) dengan operator *Sobel* dan *threshold* bernilai 1:



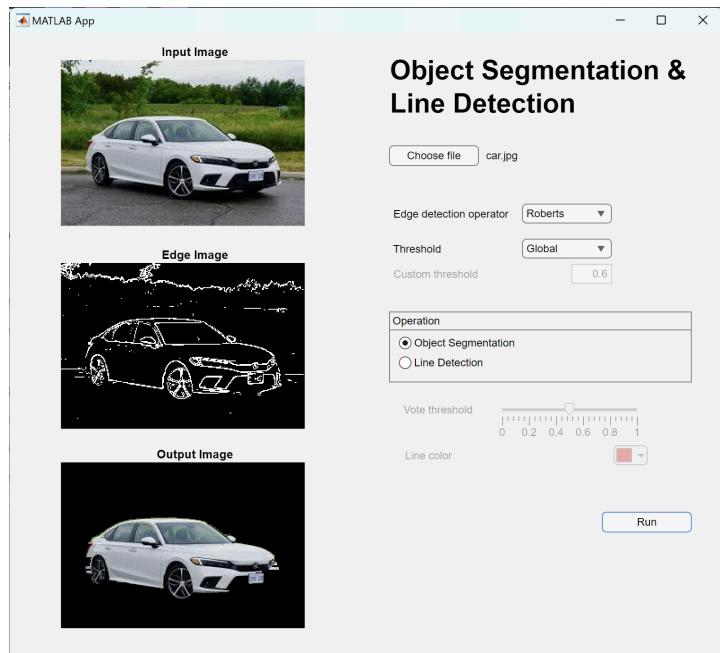
Gambar 2.3 Segmentasi objek citra mobil dengan operator Sobel

Segmentasi objek citra mobil (*car.jpg*) dengan operator *Prewitt* dan *threshold* bernilai 0.6:



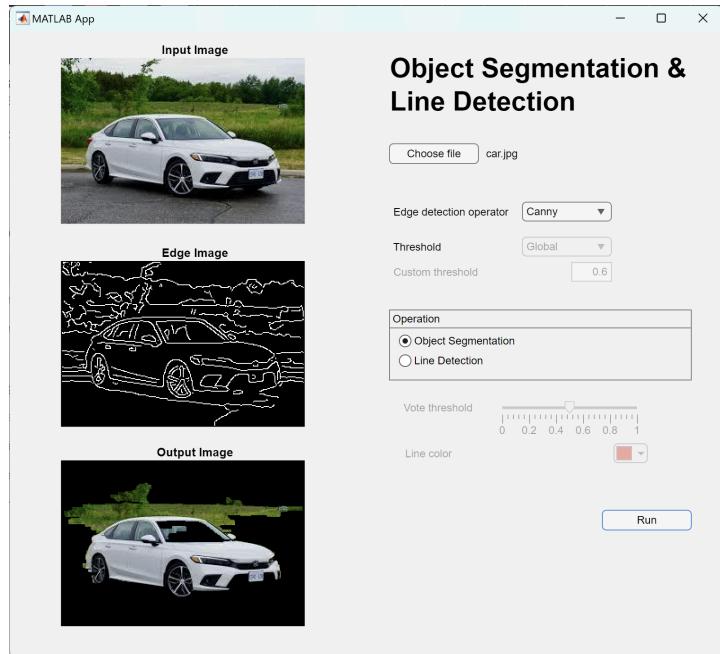
Gambar 2.4 Segmentasi objek citra mobil dengan operator *Prewitt*

Segmentasi objek citra mobil (*car.jpg*) dengan operator *Roberts* dan *threshold* bertipe global:



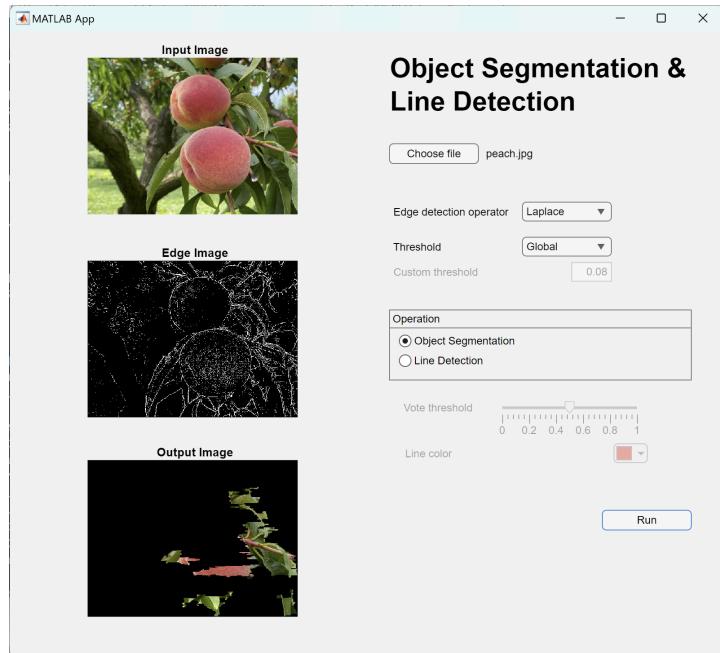
Gambar 2.5 Segmentasi objek citra mobil dengan operator *Roberts*

Segmentasi objek citra mobil (*car.jpg*) dengan operator *Canny*:



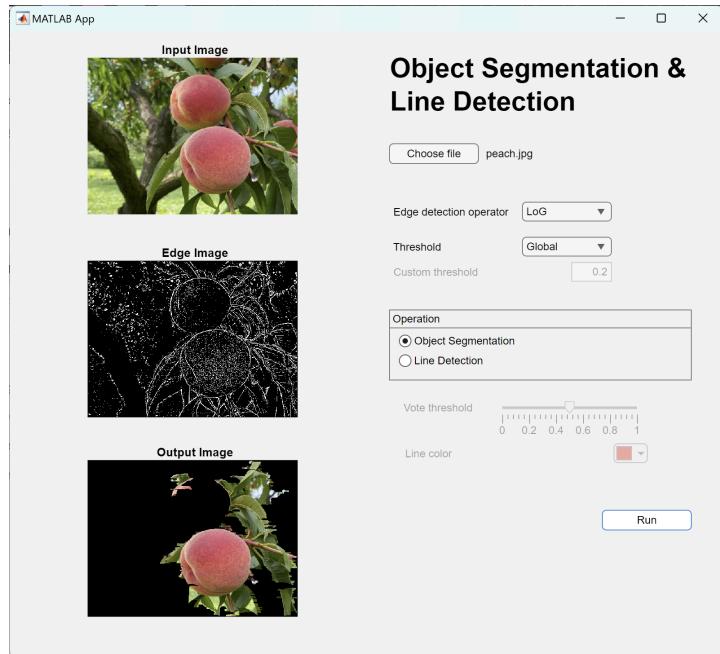
Gambar 2.6 Segmentasi objek citra mobil dengan operator *Canny*

Segmentasi objek citra buah persik (*peach.jpg*) dengan operator *Laplace* dan *threshold* bertipe global:



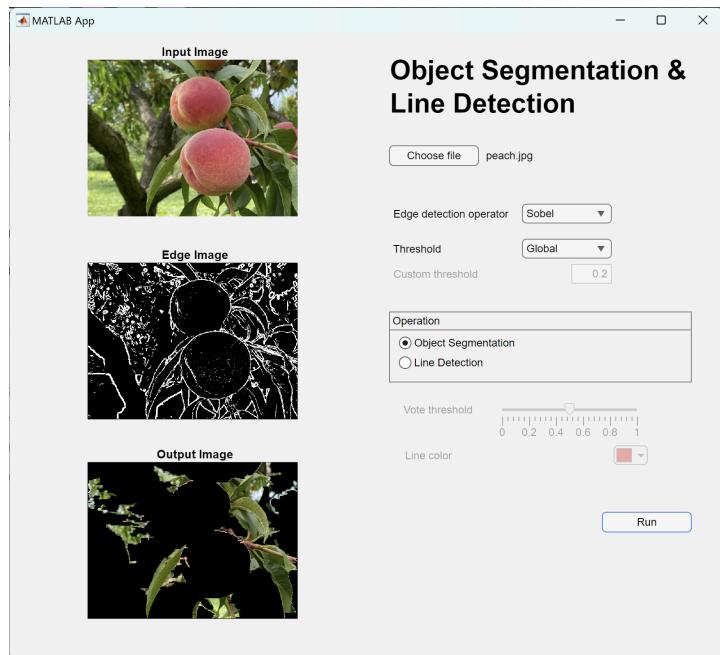
Gambar 2.7 Segmentasi objek citra buah persik dengan operator *Laplace*

Segmentasi objek citra buah persik (*peach.jpg*) dengan operator *Laplace of Gaussian* dan *threshold* bertipe global:



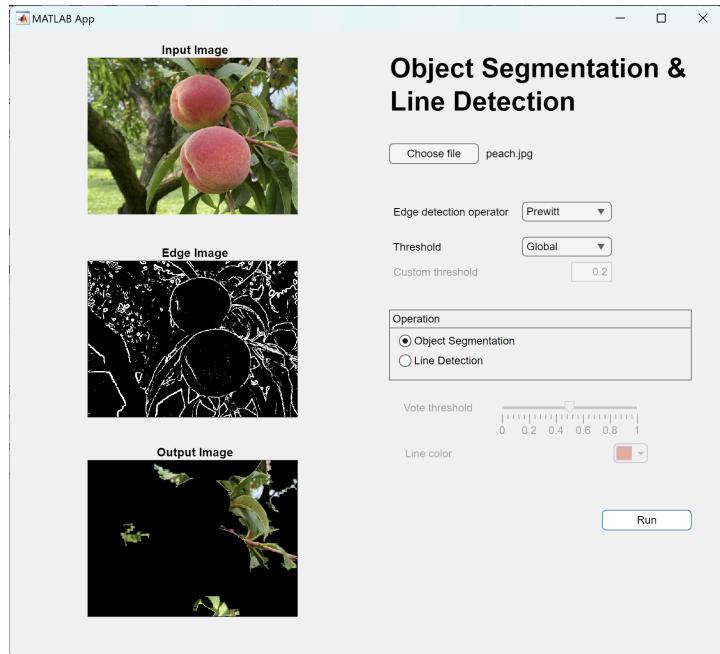
Gambar 2.8 Segmentasi objek citra buah persik dengan operator *Laplace of Gaussian*

Segmentasi objek citra buah persik (*peach.jpg*) dengan operator *Sobel* dan *threshold* bertipe global:



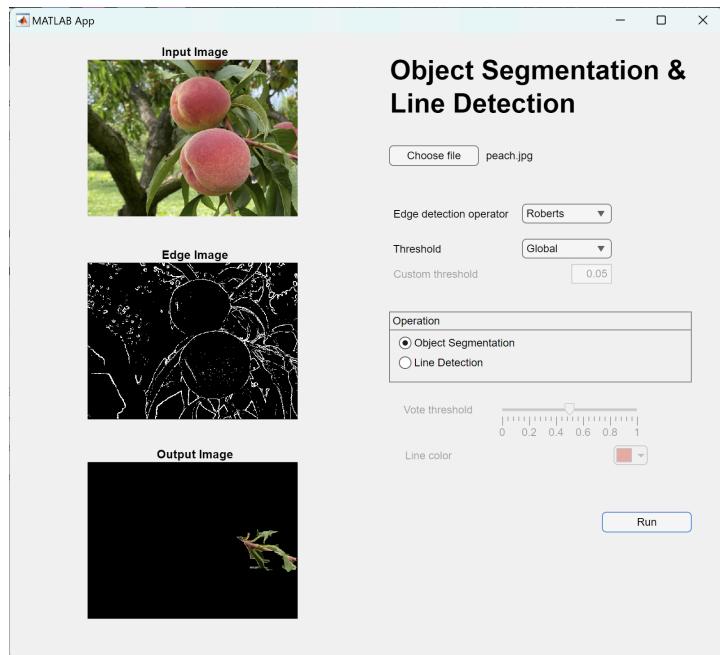
Gambar 2.9 Segmentasi objek citra buah persik dengan operator Sobel

Segmentasi objek citra buah persik (*peach.jpg*) dengan operator *Prewitt* dan *threshold* bertipe global:



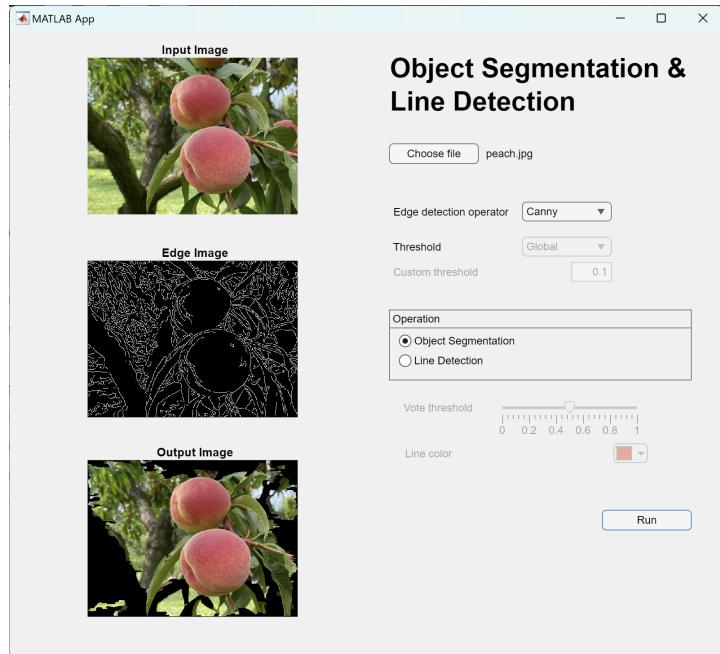
Gambar 2.10 Segmentasi objek citra buah persik dengan operator *Prewitt*

Segmentasi objek citra buah persik (*peach.jpg*) dengan operator *Roberts* dan *threshold* bertipe global:



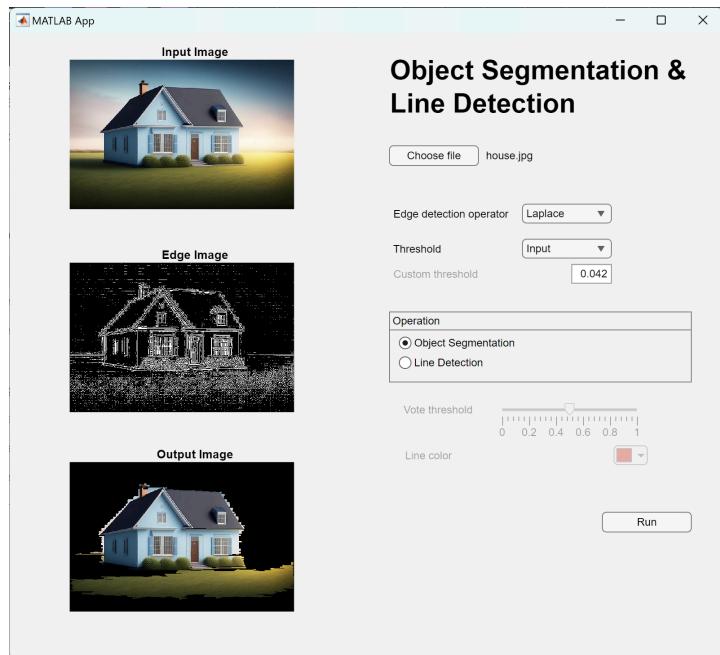
Gambar 2.11 Segmentasi objek citra buah persik dengan operator *Roberts*

Segmentasi objek citra buah persik (*peach.jpg*) dengan operator *Canny*:



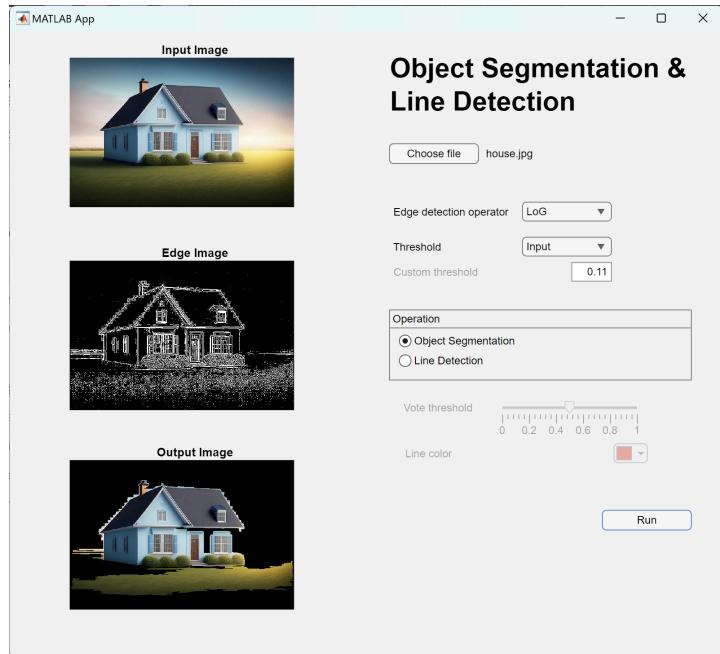
Gambar 2.12 Segmentasi objek citra buah persik dengan operator *Canny*

Segmentasi objek citra rumah (*house.jpg*) dengan operator *Laplace* dan *threshold* bernilai 0.042:



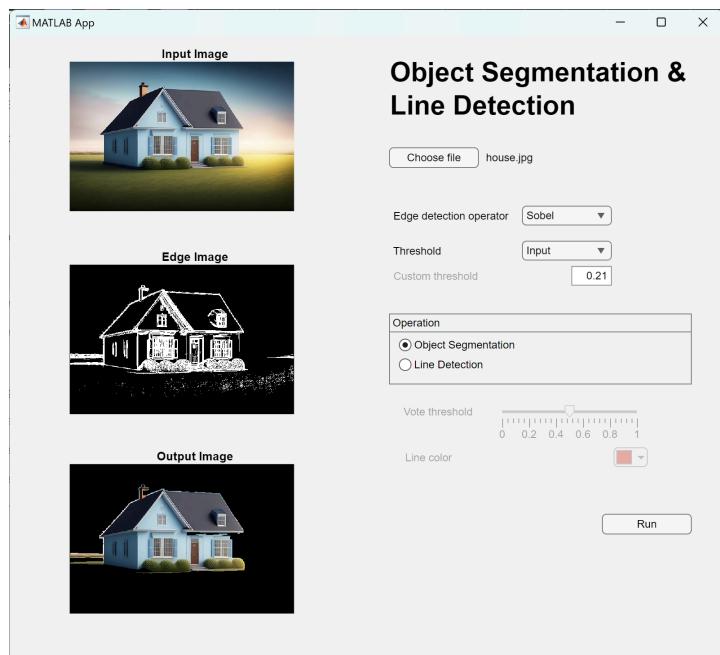
Gambar 2.13 Segmentasi objek citra rumah dengan operator *Laplace*

Segmentasi objek citra rumah (*house.jpg*) dengan operator *Laplace of Gaussian* dan *threshold* bernilai 0.11:



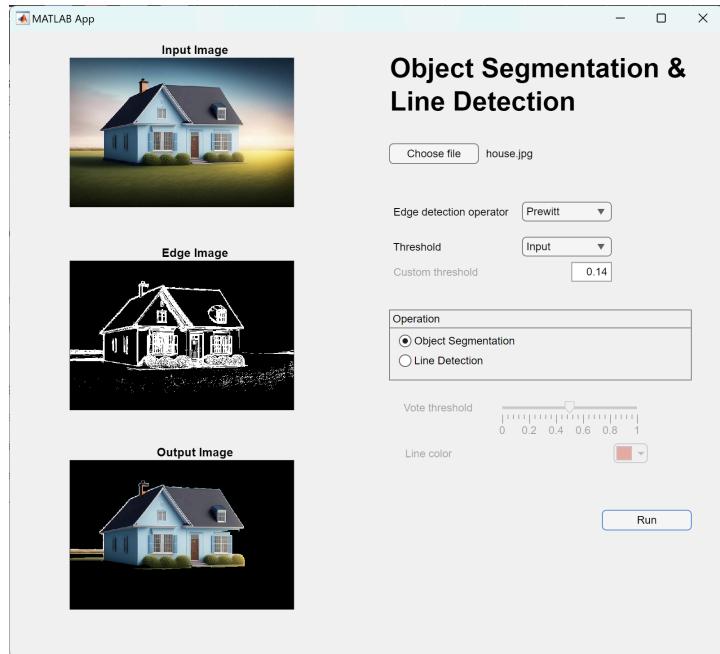
Gambar 2.14 Segmentasi objek citra rumah dengan operator *Laplace of Gaussian*

Segmentasi objek citra rumah (*house.jpg*) dengan operator *Sobel* dan *threshold* bernilai 0.21:



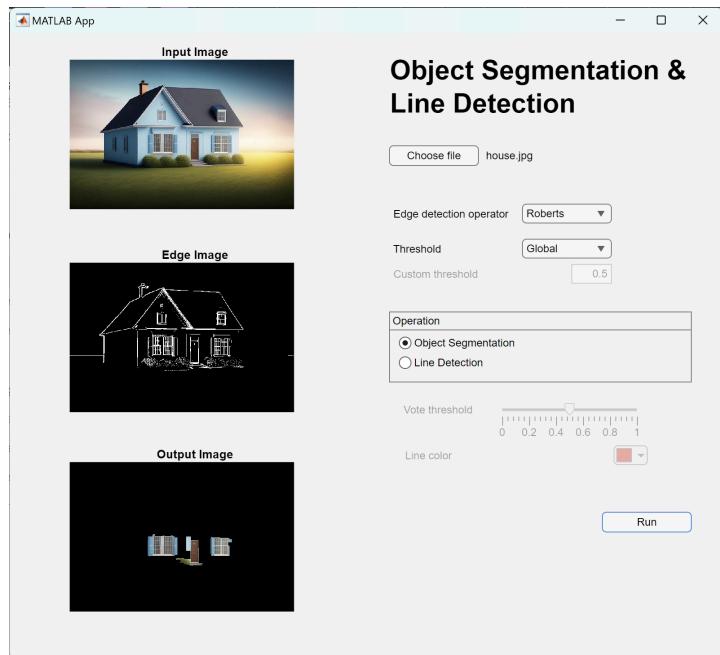
Gambar 2.15 Segmentasi objek citra rumah dengan operator Sobel

Segmentasi objek citra rumah (*house.jpg*) dengan operator *Prewitt* dan *threshold* bernilai 0.14:



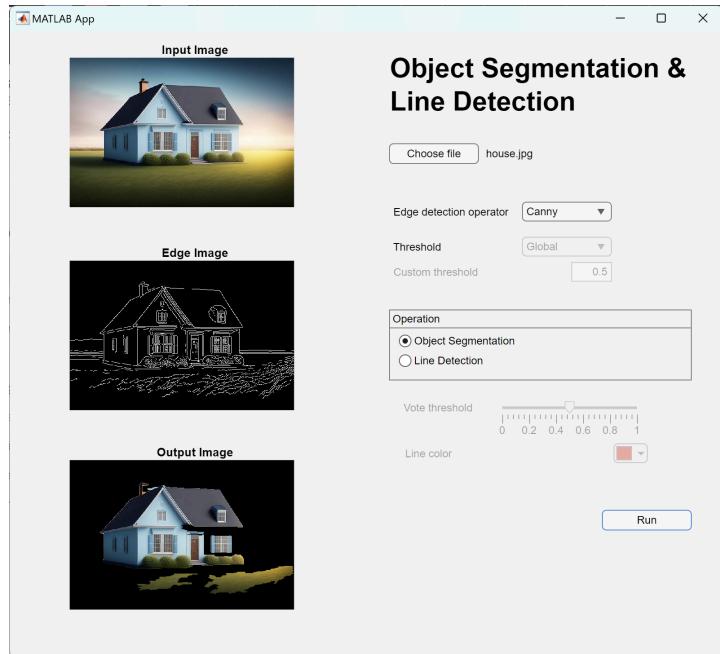
Gambar 2.16 Segmentasi objek citra rumah dengan operator *Prewitt*

Segmentasi objek citra rumah (*house.jpg*) dengan operator *Roberts* dan *threshold* bertipe global:



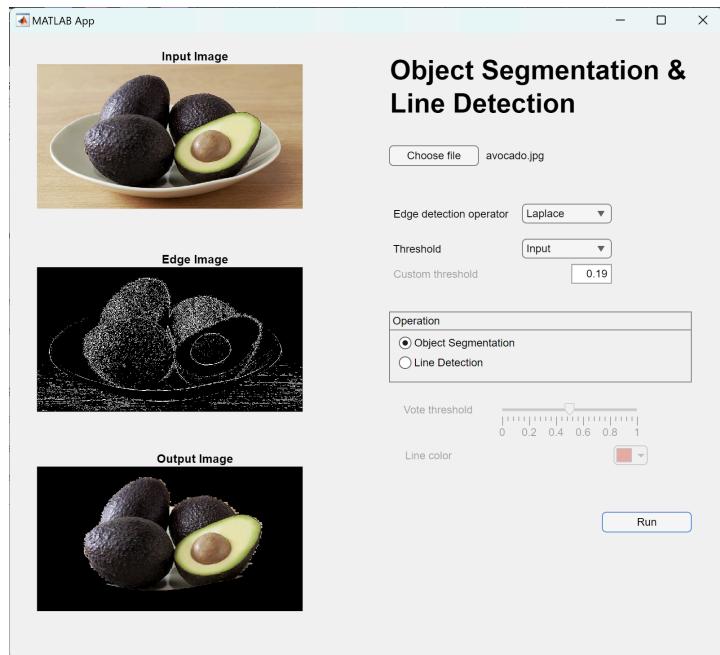
Gambar 2.17 Segmentasi objek citra rumah dengan operator *Roberts*

Segmentasi objek citra rumah (*house.jpg*) dengan operator *Canny*:



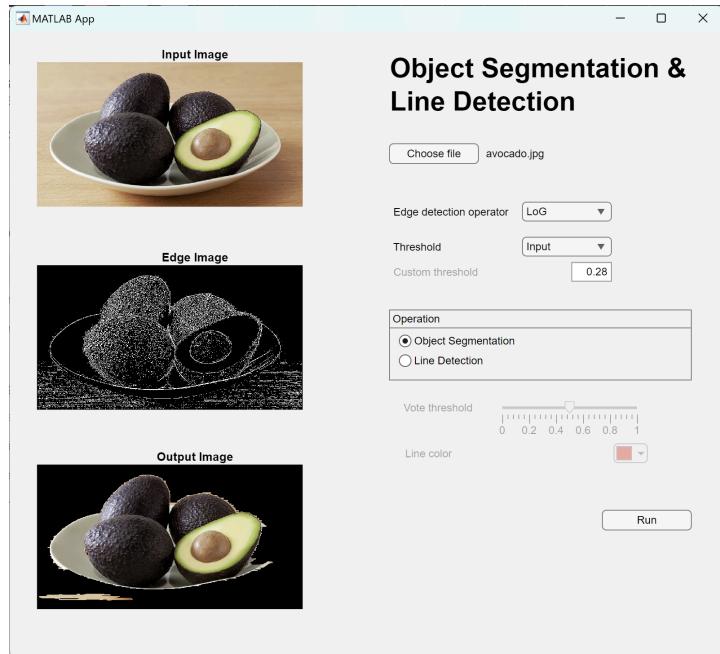
Gambar 2.18 Segmentasi objek citra rumah dengan operator *Canny*

Segmentasi objek citra alpukat (*avocado.jpg*) dengan operator *Laplace* dan *threshold* bernilai 0.19:



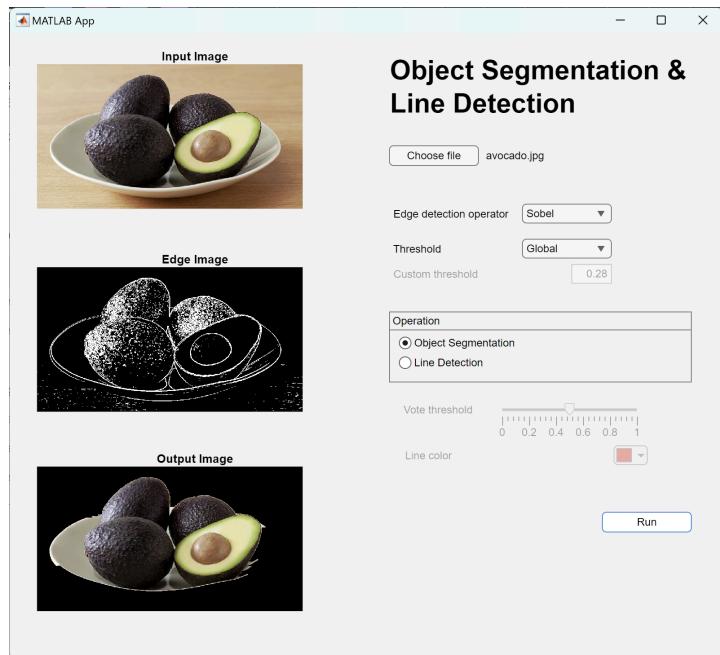
Gambar 2.19 Segmentasi objek citra alpukat dengan operator *Laplace*

Segmentasi objek citra alpukat (*avocado.jpg*) dengan operator *Laplace of Gaussian* dan *threshold* bernilai 0.28:



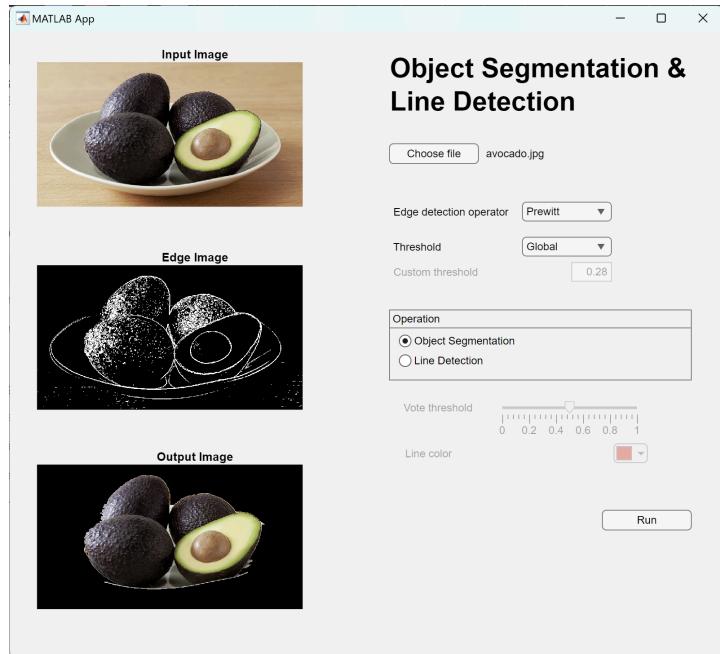
Gambar 2.20 Segmentasi objek citra alpukat dengan operator *Laplace of Gaussian*

Segmentasi objek citra alpukat (*avocado.jpg*) dengan operator *Sobel* dan *threshold* bertipe global:



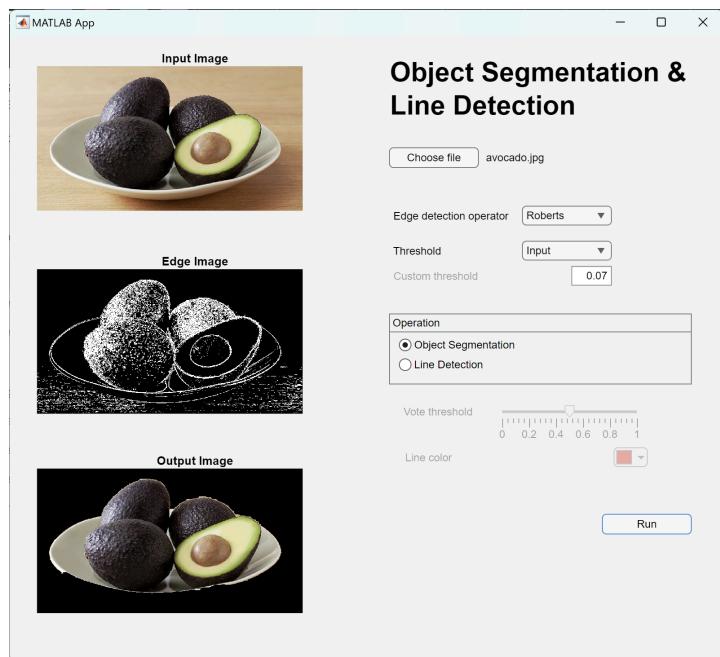
Gambar 2.21 Segmentasi objek citra alpukat dengan operator Sobel

Segmentasi objek citra alpukat (*avocado.jpg*) dengan operator *Prewitt* dan *threshold* bertipe global:



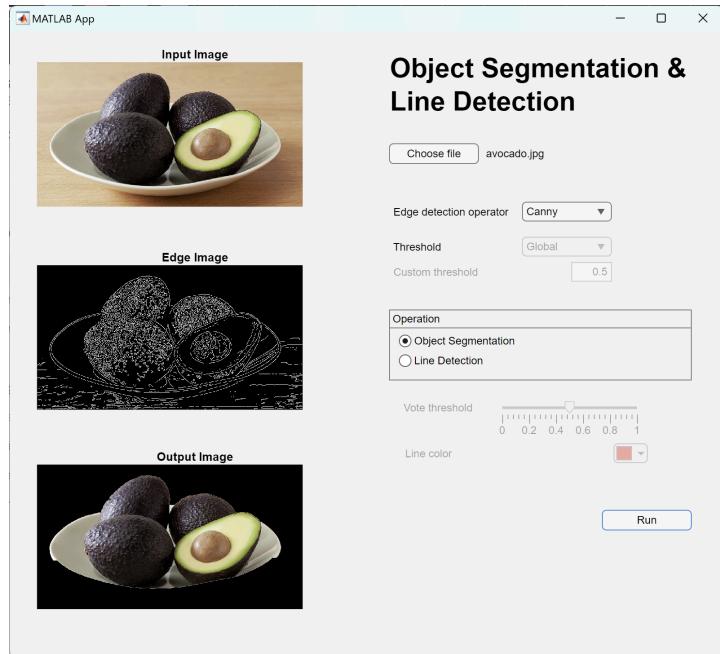
Gambar 2.22 Segmentasi objek citra alpukat dengan operator *Prewitt*

Segmentasi objek citra alpukat (*avocado.jpg*) dengan operator *Roberts* dan *threshold* bernilai 0.07:



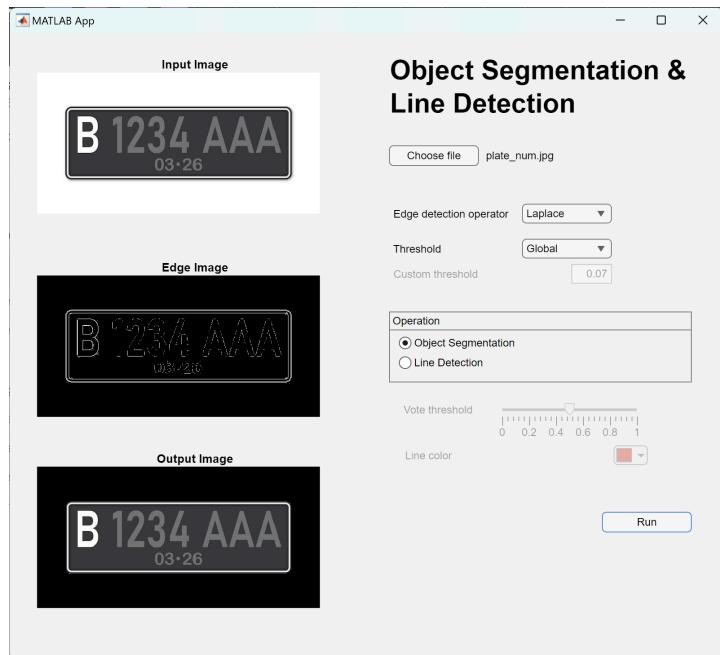
Gambar 2.23 Segmentasi objek citra alpukat dengan operator *Roberts*

Segmentasi objek citra alpukat (*avocado.jpg*) dengan operator *Canny*:



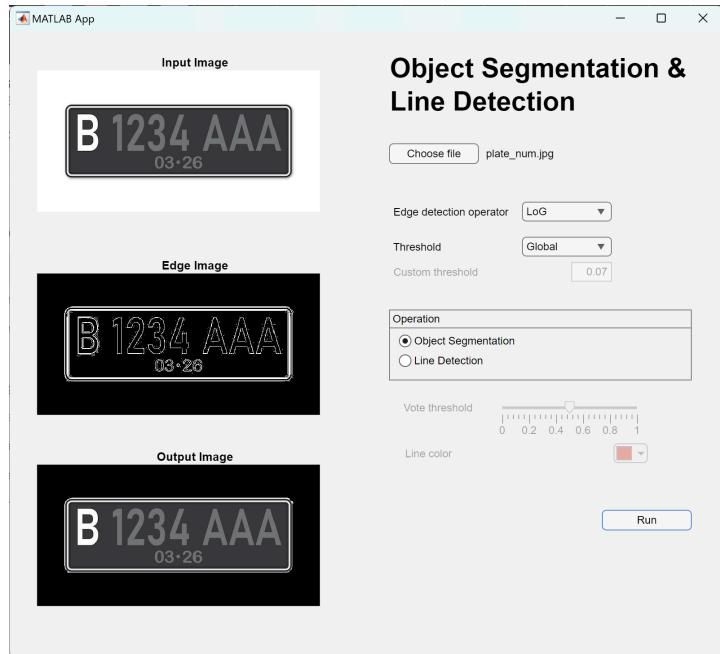
Gambar 2.24 Segmentasi objek citra alpukat dengan operator *Canny*

Segmentasi objek citra pelat nomor kendaraan (*plate_num.jpg*) dengan operator *Laplace* dan *threshold* bertipe global:



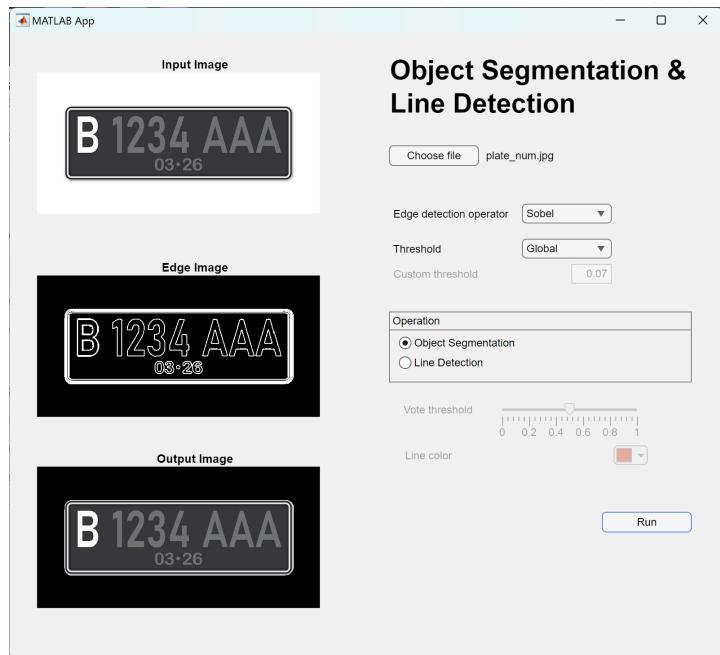
Gambar 2.25 Segmentasi objek citra pelat nomor kendaraan dengan operator *Laplace*

Segmentasi objek citra pelat nomor kendaraan (*plate_num.jpg*) dengan operator *Laplace of Gaussian* dan *threshold* bertipe global:



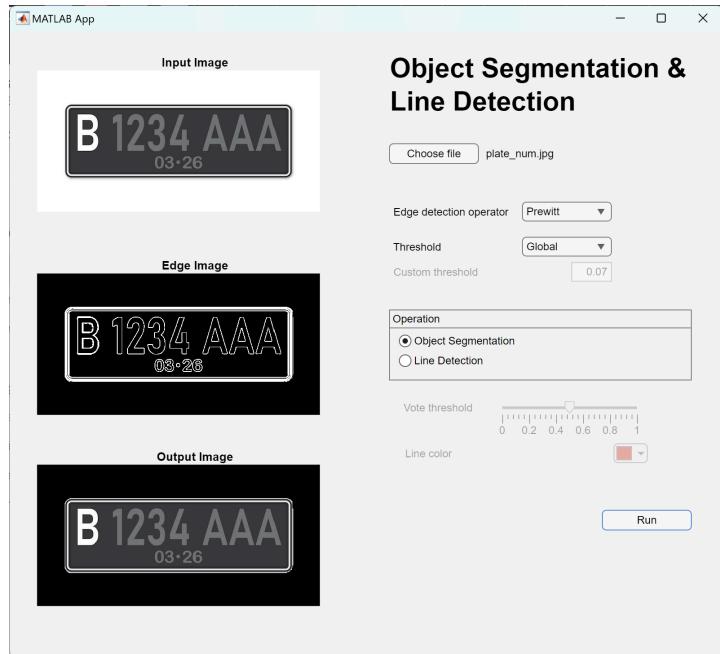
Gambar 2.26 Segmentasi objek citra pelat nomor kendaraan dengan operator *Laplace of Gaussian*

Segmentasi objek citra pelat nomor kendaraan (*plate_num.jpg*) dengan operator *Sobel* dan *threshold* bertipe global:



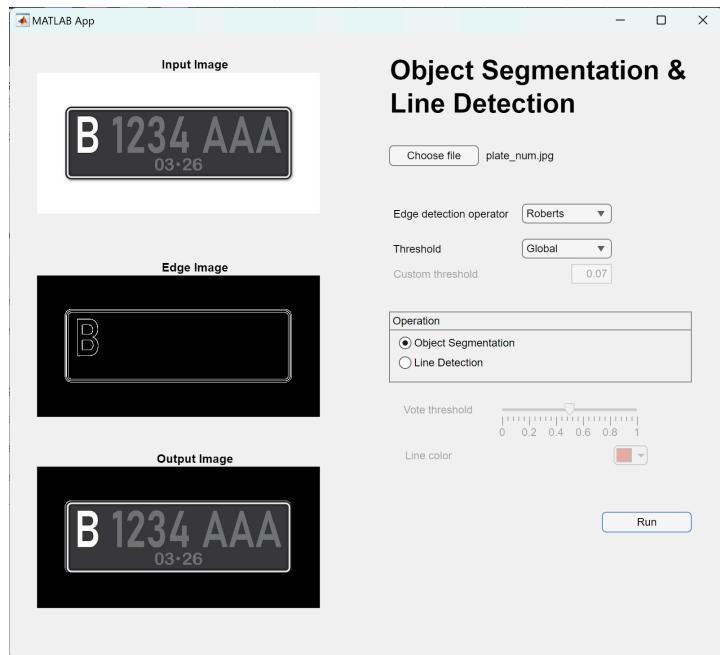
Gambar 2.27 Segmentasi objek citra pelat nomor kendaraan dengan operator Sobel

Segmentasi objek citra pelat nomor kendaraan (*plate_num.jpg*) dengan operator *Prewitt* dan *threshold* bertipe global:



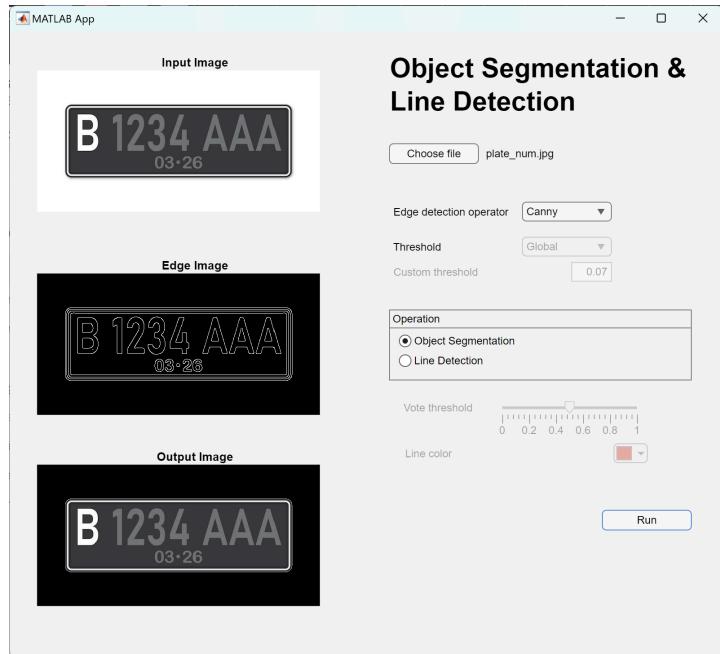
Gambar 2.28 Segmentasi objek citra pelat nomor kendaraan dengan operator *Prewitt*

Segmentasi objek citra pelat nomor kendaraan (*plate_num.jpg*) dengan operator *Roberts* dan *threshold* bertipe global:



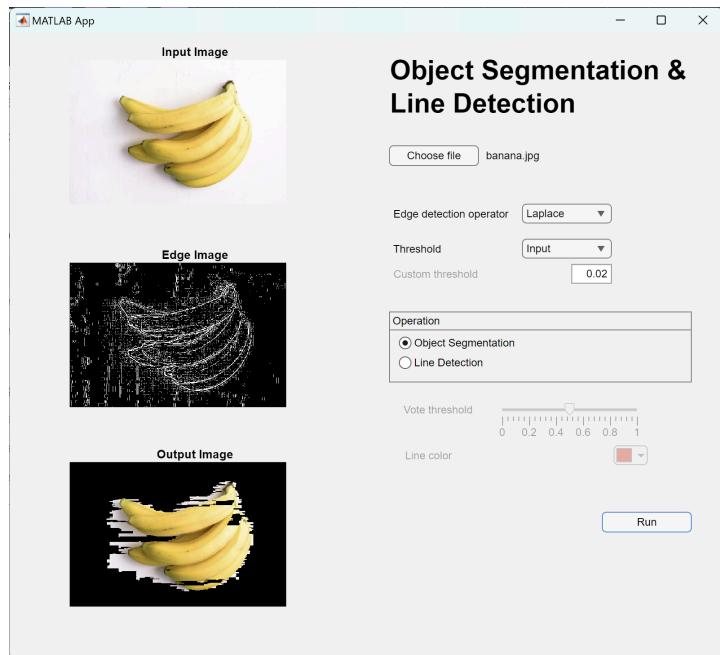
Gambar 2.29 Segmentasi objek citra pelat nomor kendaraan dengan operator *Roberts*

Segmentasi objek citra pelat nomor kendaraan (*plate_num.jpg*) dengan operator *Canny*:



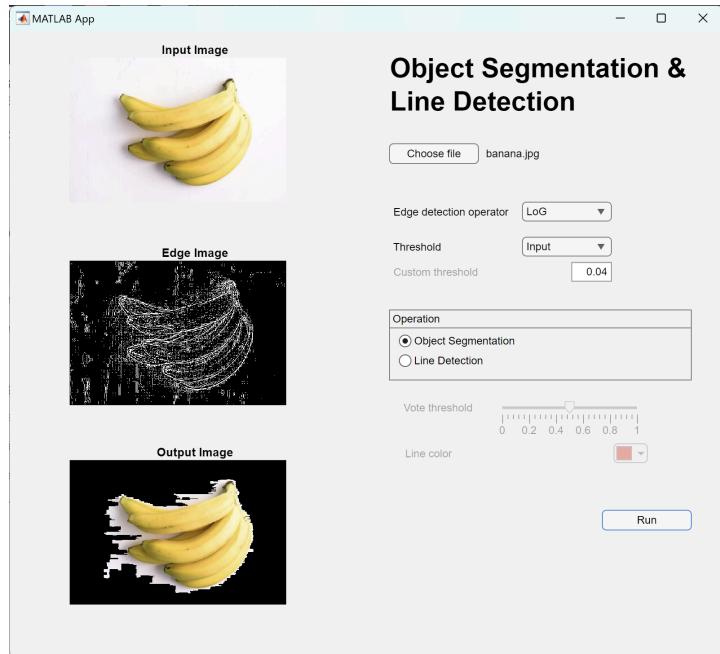
Gambar 2.30 Segmentasi objek citra pelat nomor kendaraan dengan operator *Canny*

Segmentasi objek citra pisang (*banana.jpg*) dengan operator *Laplace* dan *threshold* bernilai 0.02:



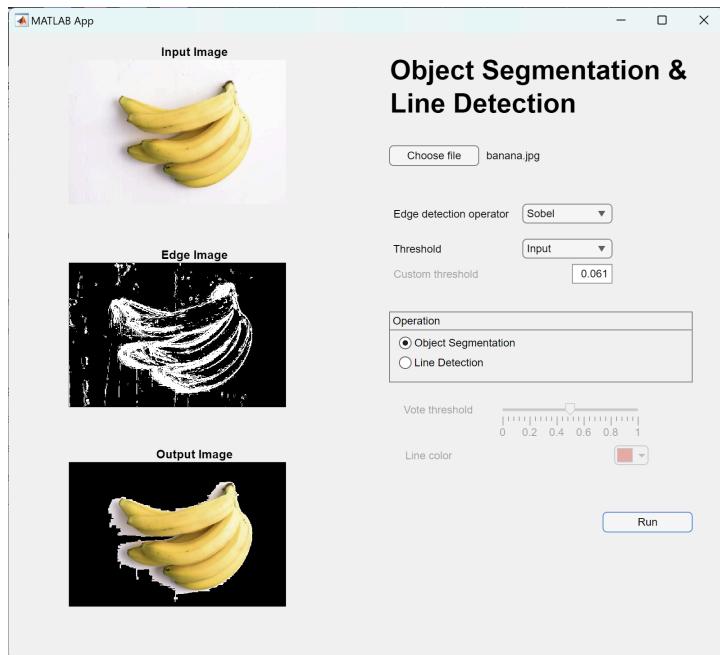
Gambar 2.31 Segmentasi objek citra pisang dengan operator *Laplace*

Segmentasi objek citra pisang (*banana.jpg*) dengan operator *Laplace of Gaussian* dan *threshold* bernilai 0.04:



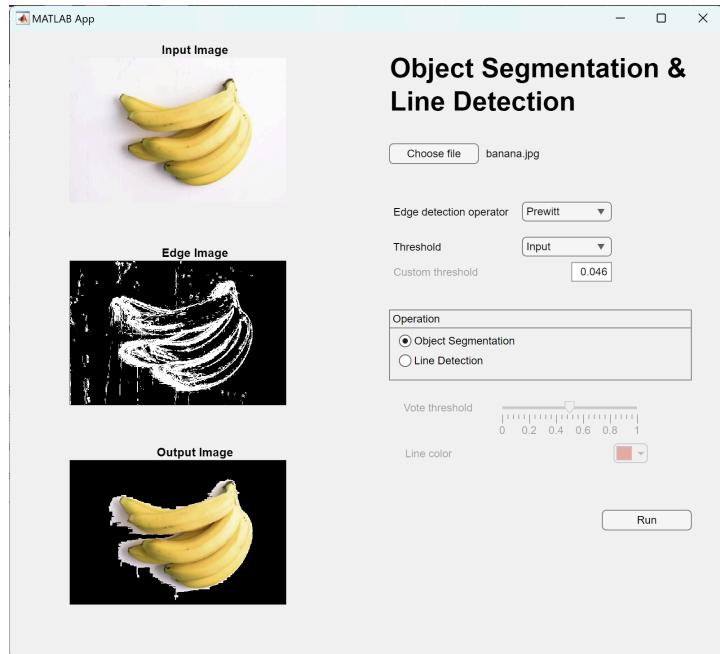
Gambar 2.32 Segmentasi objek citra pisang dengan operator *Laplace of Gaussian*

Segmentasi objek citra pisang (*banana.jpg*) dengan operator *Sobel* dan *threshold* bernilai 0.061:



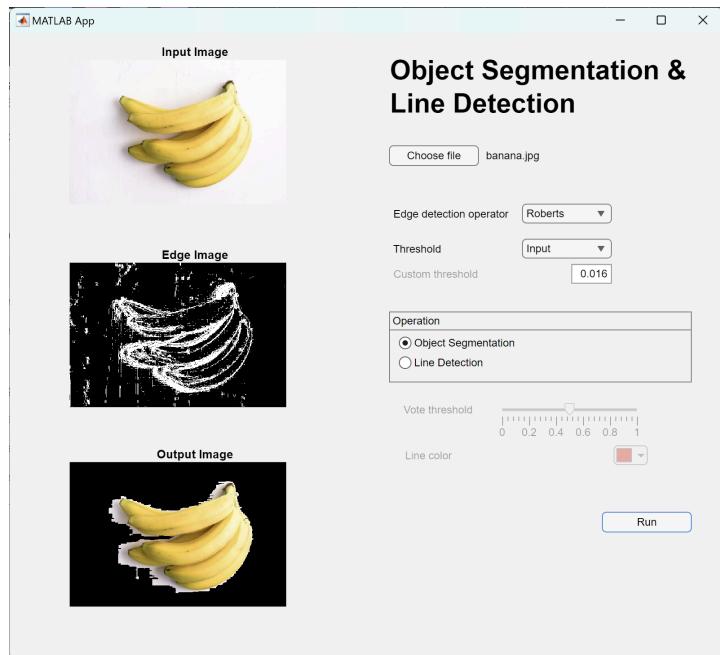
Gambar 2.33 Segmentasi objek citra pisang dengan operator Sobel

Segmentasi objek citra pisang (*banana.jpg*) dengan operator *Prewitt* dan *threshold* bernilai 0.046:



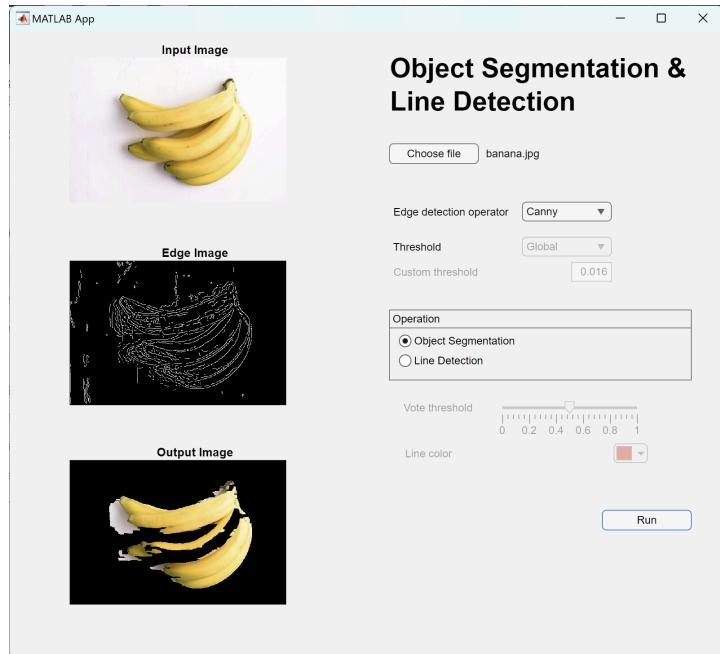
Gambar 2.34 Segmentasi objek citra pisang dengan operator *Prewitt*

Segmentasi objek citra pisang (*banana.jpg*) dengan operator *Roberts* dan *threshold* bernilai 0.016:



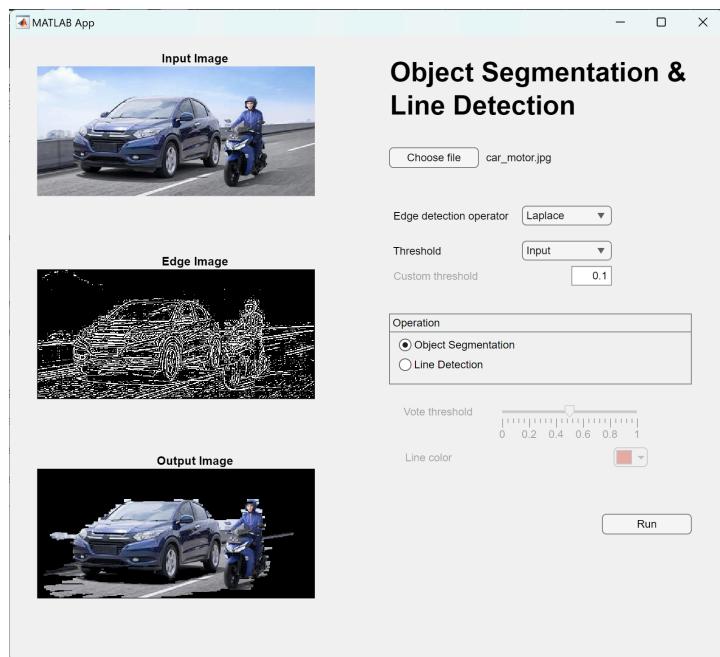
Gambar 2.35 Segmentasi objek citra pisang dengan operator *Roberts*

Segmentasi objek citra pisang (*banana.jpg*) dengan operator *Canny*:



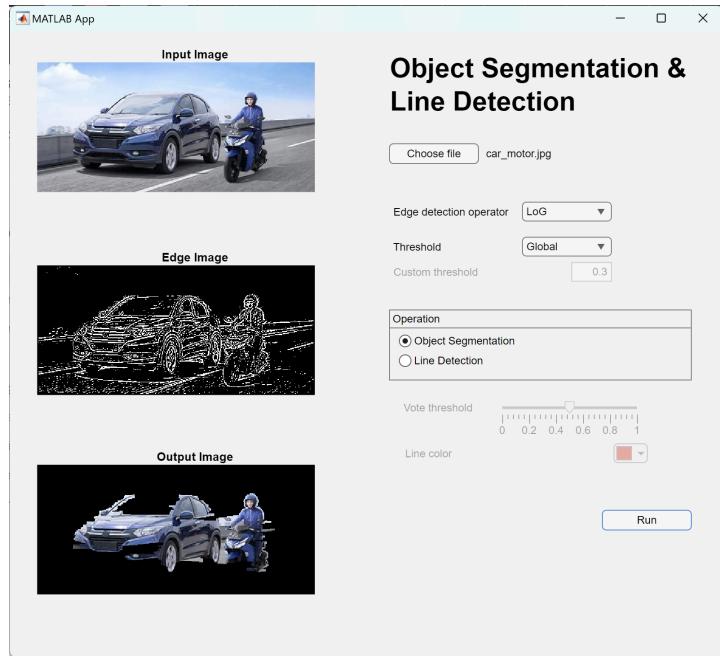
Gambar 2.36 Segmentasi objek citra pisang dengan operator *Canny*

Segmentasi objek citra mobil dan motor (*car_motor.jpg*) dengan operator *Laplace* dan *threshold* bernilai 0.1:



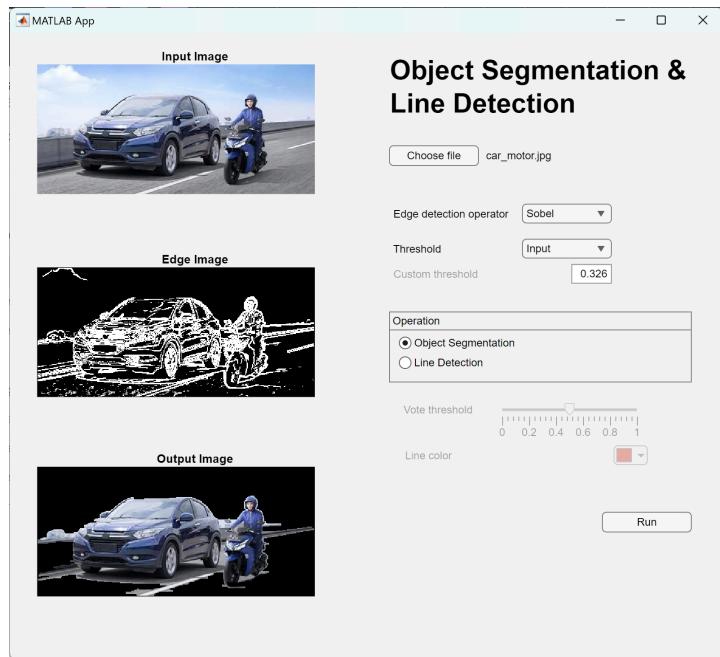
Gambar 2.37 Segmentasi objek citra mobil dan motor dengan operator *Laplace*

Segmentasi objek citra mobil dan motor (*car_motor.jpg*) dengan operator *Laplace of Gaussian* dan *threshold* bertipe global:



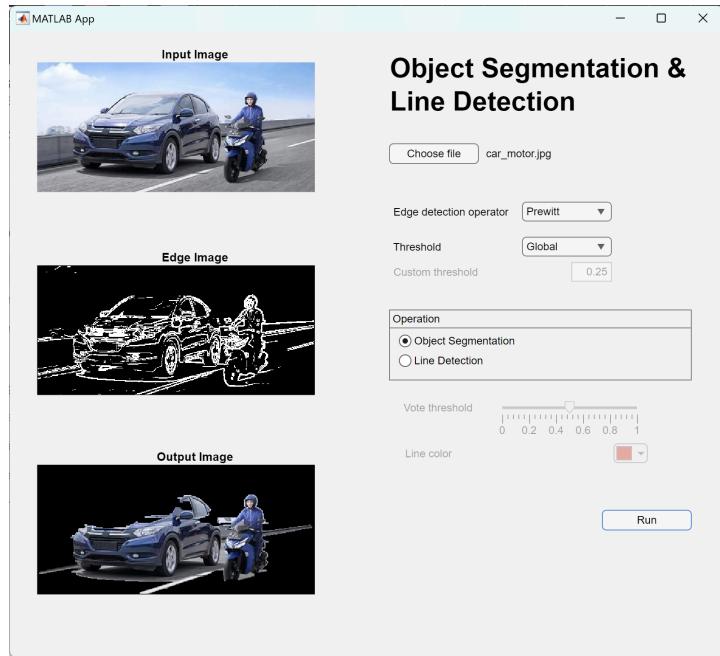
Gambar 2.38 Segmentasi objek citra mobil dan motor dengan operator *Laplace of Gaussian*

Segmentasi objek citra mobil dan motor (*car_motor.jpg*) dengan operator *Sobel* dan *threshold* bernilai 0.326:



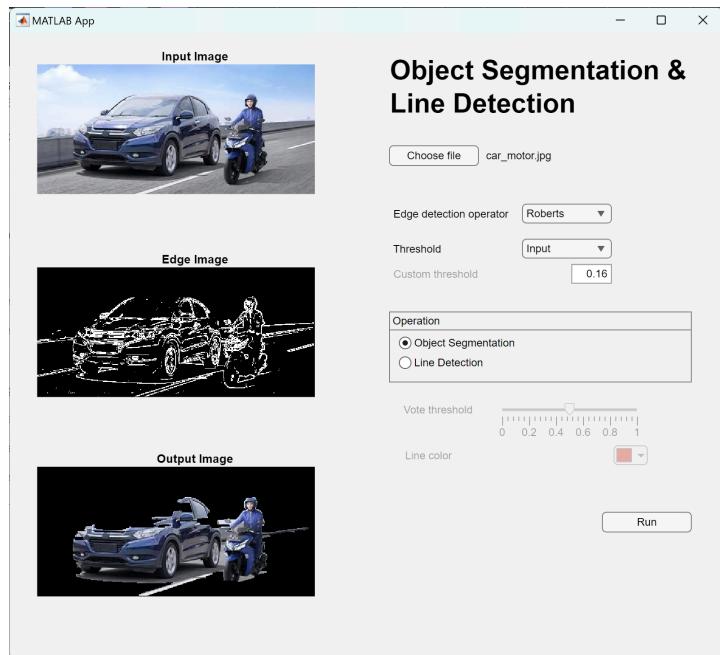
Gambar 2.39 Segmentasi objek citra mobil dan motor dengan operator Sobel

Segmentasi objek citra mobil dan motor (*car_motor.jpg*) dengan operator *Prewitt* dan *threshold* bertipe global:



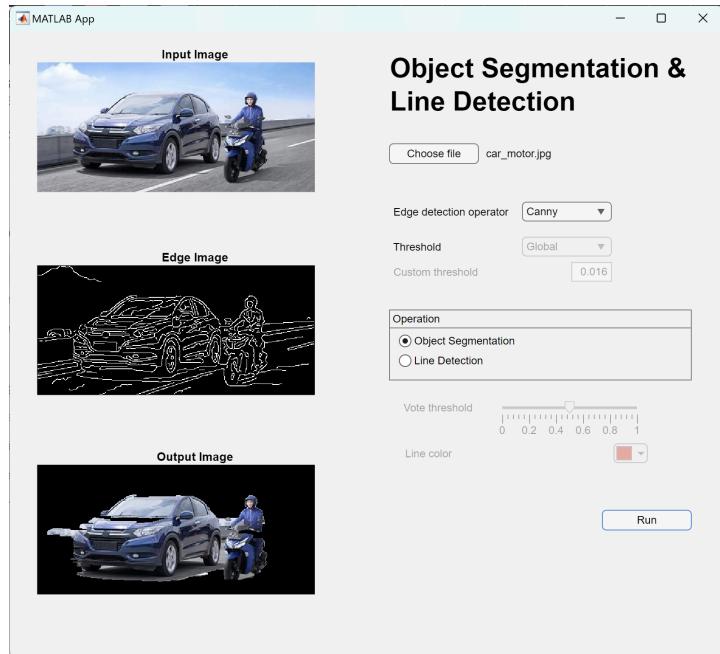
Gambar 2.40 Segmentasi objek citra mobil dan motor dengan operator *Prewitt*

Segmentasi objek citra mobil dan motor (*car_motor.jpg*) dengan operator *Roberts* dan *threshold* bernilai 0.16:



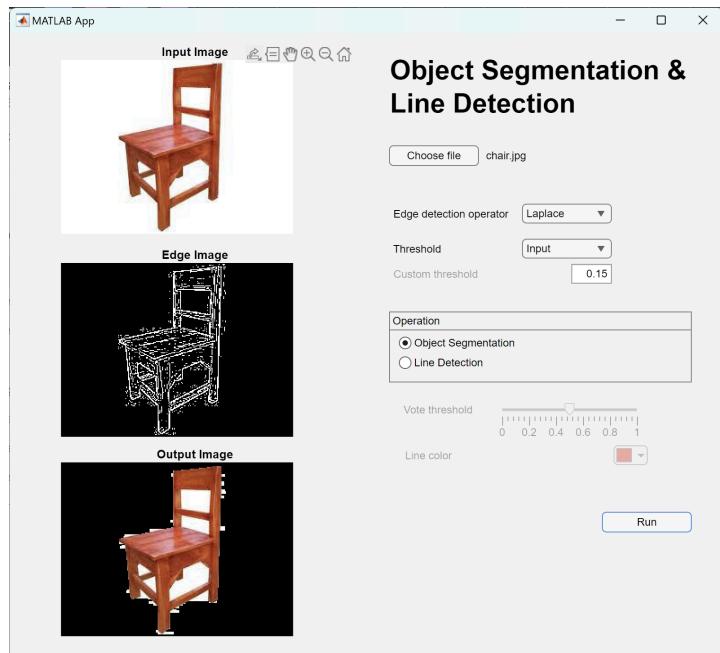
Gambar 2.41 Segmentasi objek citra mobil dan motor dengan operator *Roberts*

Segmentasi objek citra mobil dan motor (*car_motor.jpg*) dengan operator *Canny*:



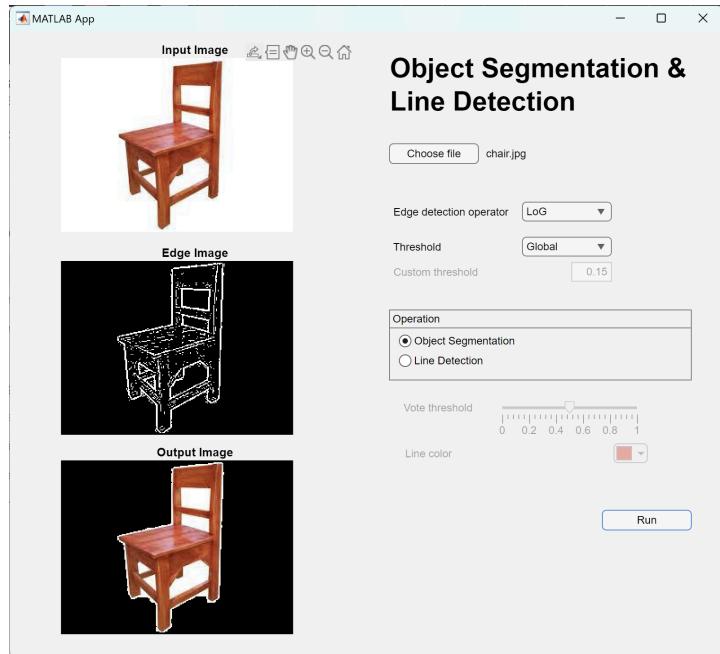
Gambar 2.42 Segmentasi objek citra mobil dan motor dengan operator *Canny*

Segmentasi objek citra kursi (*chair.jpg*) dengan operator *Laplace* dan *threshold* bernilai 0.15:



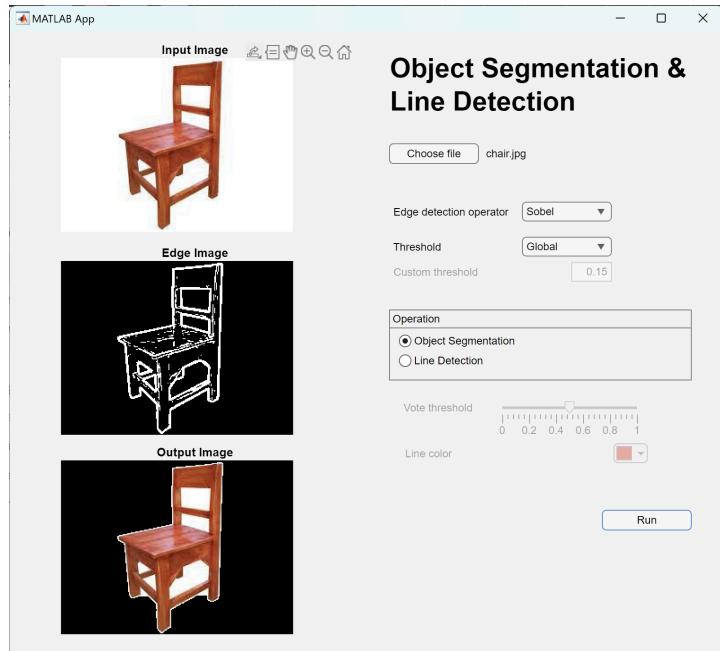
Gambar 2.43 Segmentasi objek citra kursi dengan operator *Laplace*

Segmentasi objek citra kursi (*chair.jpg*) dengan operator *Laplace of Gaussian* dan *threshold* bertipe global:



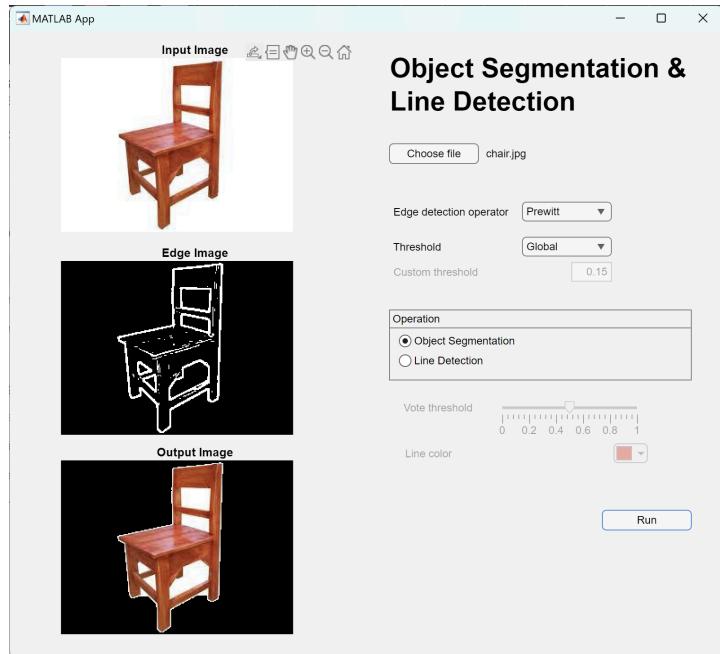
Gambar 2.44 Segmentasi objek citra kursi dengan operator *Laplace of Gaussian*

Segmentasi objek citra kursi (*chair.jpg*) dengan operator *Sobel* dan *threshold* bertipe global:



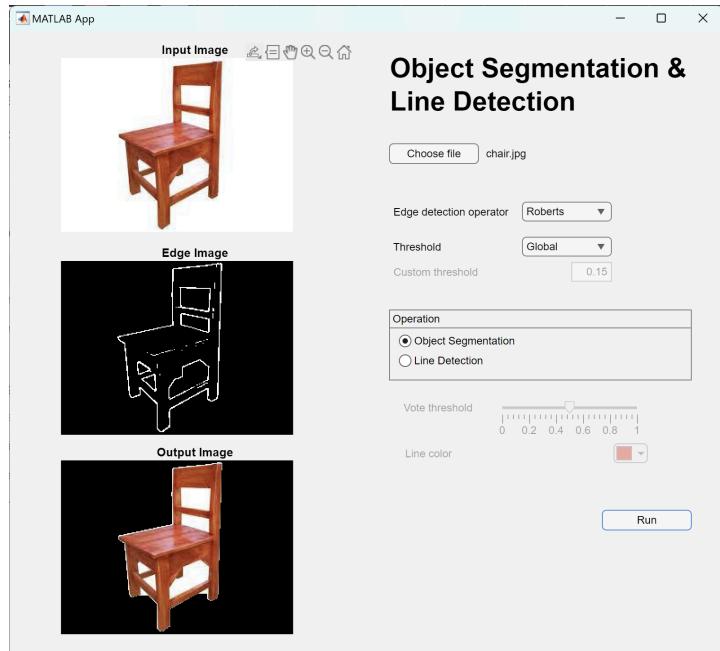
Gambar 2.45 Segmentasi objek citra kursi dengan operator Sobel

Segmentasi objek citra kursi (*chair.jpg*) dengan operator *Prewitt* dan *threshold* bertipe global:



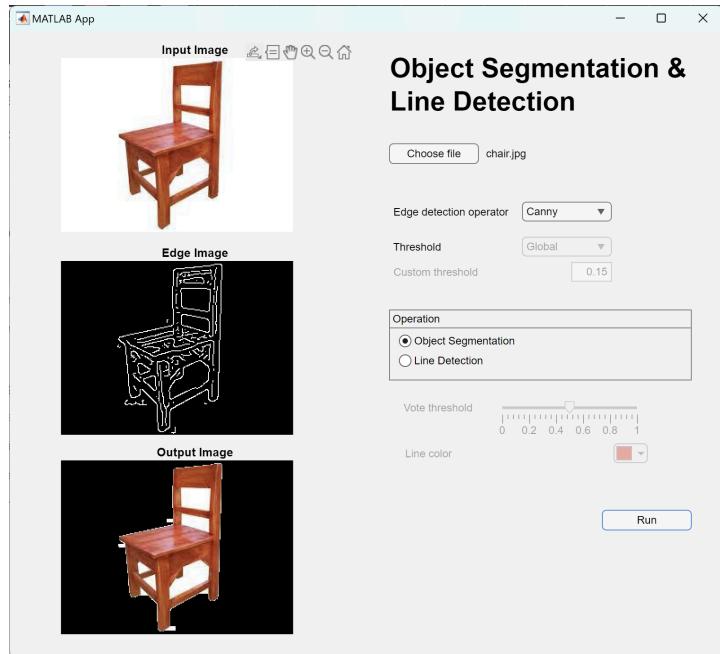
Gambar 2.46 Segmentasi objek citra kursi dengan operator *Prewitt*

Segmentasi objek citra kursi (*chair.jpg*) dengan operator *Roberts* dan *threshold* bertipe global:



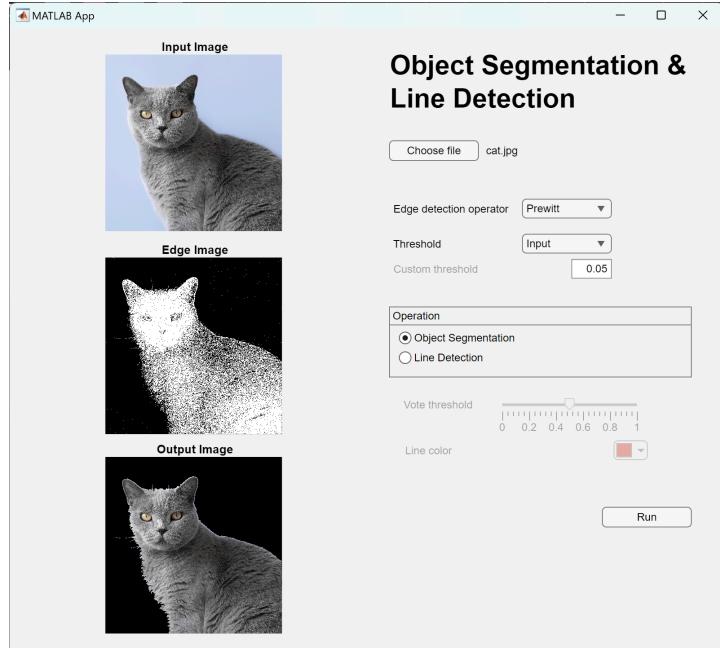
Gambar 2.47 Segmentasi objek citra kursi dengan operator *Roberts*

Segmentasi objek citra kursi (*chair.jpg*) dengan operator *Canny*:



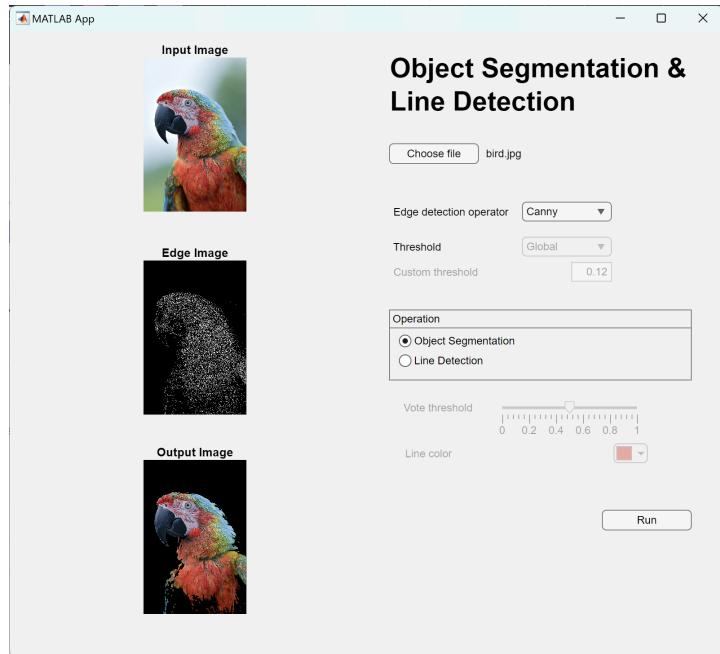
Gambar 2.48 Segmentasi objek citra kursi dengan operator *Canny*

Segmentasi objek citra kucing (*cat.jpg*) dengan operator *Prewitt* dan *threshold* bernilai 0.05:



Gambar 2.49 Segmentasi objek citra kucing dengan operator *Prewitt*

Segmentasi objek citra burung (*bird.jpg*) dengan operator *Canny*:



Gambar 2.50 Segmentasi objek citra burung dengan operator *Canny*

Analisis hasil:

Tidak semua citra yang dihasilkan dari segmentasi objek menghasilkan segmentasi yang bagus. Hal ini disebabkan karena deteksi tepi yang dihasilkan tidak sempurna, yaitu terdapat garis tepi yang tidak terdeteksi atau tepi pada latar belakang terdeteksi sehingga tidak terfokus pada objeknya. Hal ini dipengaruhi oleh operator (metode) deteksi tepi yang digunakan karena setiap metode memiliki karakteristik dan sensitivitas yang berbeda terhadap perubahan intensitas piksel.

Operator *Laplace* menggunakan operator berbasis derivatif kedua untuk mendeteksi perubahan intensitas yang tajam. Namun, *Laplace* sangat sensitif terhadap derau, sehingga hasilnya seringkali memiliki tepi tambahan yang tidak diinginkan.

Operator *Laplace of Gaussian* (LoG) menggabungkan penghalusan menggunakan filter *Gaussian* dengan deteksi tepi *Laplace*. Ini mengurangi sensitivitas terhadap derau dibandingkan *Laplace* murni, tetapi masih mungkin menghasilkan tepi yang kabur jika parameter *Gaussian* tidak tepat.

Operator *Sobel* mendeteksi tepi berdasarkan gradien pertama dalam arah horizontal dan vertikal. *Sobel* lebih halus terhadap derau dibandingkan *Laplace* dan cocok

untuk mendeteksi tepi yang lebih tegas, tetapi terkadang gagal mendeteksi tepi yang sangat tipis.

Operator *Prewitt* mirip dengan *Sobel*, tetapi dengan bobot kernel yang lebih sederhana. Ini sering menghasilkan tepi yang kurang tajam dibandingkan *Sobel* dan lebih sensitif terhadap derau

Operator *Roberts* mendeteksi tepi pada arah diagonal dengan menggunakan *kernel* kecil (2x2). Karena ukurannya kecil, *Roberts* sangat sensitif terhadap derau dan sering menghasilkan hasil yang lebih kasar dibandingkan *Sobel* atau *Prewitt*.

Operator *Canny* lebih kompleks dan mencakup penghalusan dengan *Gaussian*, perhitungan gradien, *non-maximum suppression*, dan *thresholding*. Metode ini biasanya memberikan hasil deteksi tepi yang paling bersih dan terfokus pada objek, tetapi lebih lambat karena kompleksitasnya.

Selain jenis operator, hasil segmentasi objek juga ditentukan oleh teknik pengambangan (*thresholding*) yang digunakan. Terdapat empat jenis teknik pengambangan yang dapat digunakan, yaitu global, *adaptive*, Otsu, dan nilai pengambangan.

Pengambangan global menggunakan nilai intensitas tetap untuk seluruh citra (global), sehingga efektif untuk citra dengan pencahayaan seragam, tetapi kurang baik jika terdapat variasi intensitas pada latar belakang.

Pengambangan *adaptive* menghitung nilai *threshold* lokal berdasarkan lingkungan sekitar piksel, sehingga cocok untuk citra dengan pencahayaan tidak merata, tetapi rentan menghasilkan derau.

Pengambangan Otsu otomatis mencari nilai *threshold* optimal berdasarkan distribusi histogram, sehingga cocok untuk memisahkan latar depan dan latar belakang, tetapi mungkin gagal jika histogram tidak bimodal (terdapat banyak gradien di citra).

Untuk pengambangan dengan nilai tertentu, nilai *threshold* adalah di antara 0 hingga 1. Semakin besar nilai *threshold*, maka hanya bagian tepi dengan intensitas perubahan sangat tinggi yang dipertahankan, sehingga beberapa detail halus atau tepi lemah bisa hilang. Sebaliknya, semakin kecil nilai *threshold*, maka lebih banyak tepi yang terdeteksi, termasuk derau dari latar belakang yang dapat menyebabkan segmentasi menjadi kurang fokus pada objek utama.

2.2. Deteksi Jalur Garis di Jalan Raya dengan Transformasi Hough

Berikut merupakan kode program fungsi untuk pendekslsian garis lurus dalam citra.

```
function lines = lineDetection(edges)
    [M, N] = size(edges, [1 2]);
    D = sqrt(M*M + N*N);
    lines = zeros([floor(2 * D) 180]);

    for i = 1:M
        for j = 1:N
            if edges(i, j) > 0
                for k = 1:180
                    % Calculate theta and rho of every possible
                    % line that passes through (M, N)
                    theta = (k - 91) * pi / 180;
                    r = (j - 1) * cos(theta) + (i - 1) * sin(theta);

                    r = floor(r + D);
                    lines(r, k) = lines(r, k) + 1;
                end
            end
        end
    end
```

Fungsi `lineDetection` menerima parameter `edges`, yaitu hasil pendekslsian tepi dari citra masukan. Fungsi ini merupakan implementasi dari algoritma transformasi Hough. Transformasi Hough merupakan algoritma untuk mengenali bentuk-bentuk parametrik sederhana, seperti garis lurus dan lingkaran, dengan melakukan voting terhadap kombinasi nilai parameter. Dalam pendekslsian garis lurus, parameter yang digunakan adalah ρ (rho) dan θ (theta), yaitu jarak dan sudut garis terhadap titik asal, sedemikian hingga setiap titik (x, y) pada garis memenuhi persamaan parametrik berikut:

$$\rho = x \cos \theta + y \sin \theta$$

Pertama, algoritma akan terlebih dulu menginisiasi sebuah matriks P , di mana setiap elemen (i, j) dalam P merupakan akumulator untuk pasangan nilai parameter (ρ_i, θ_j) . Kemudian, untuk setiap piksel tepi, nilai ρ dan θ dihitung untuk setiap garis lurus yang mungkin yang melalui piksel tersebut. Setiap piksel tepi menyumbang suara dengan menambahkan satu kepada setiap elemen dalam P yang bersesuaian dengan parameter (ρ, θ) . Hasil dari pendekripsi garis lurus ini merupakan matriks P , di mana k puncak dari matriks bersesuaian dengan k garis lurus yang terdeteksi dalam citra.

Hasil pendekripsi garis lurus dengan transformasi Hough dapat divisualisasikan dengan melakukan transformasi Hough balikan. Berikut merupakan kode program fungsi untuk visualisasi garis lurus.

```

function drawnLines = lineDraw(lines, M, N, threshold)
D = sqrt(M*M + N*N);
drawnLines = zeros([M N]);
[R, K] = size(lines, [1 2]);

for r = 1:R
    for k = 1:K
        if lines(r, k) >= threshold
            % Compute corresponding rho and theta
            theta = (k - 1 - K/2) * pi / 180;
            rho = r - D;

            s = sin(theta);
            c = cos(theta);

            if s ~= 0
                % Draw line using DDA algorithm
                y1 = rho / s;
                y2 = (rho - (N - 1) * c) / s;
                dy = y2 - y1;

                step = max(abs(dy), N);
                incX = N / step;
                incY = dy / step;

                x = 1;
                y = y1;

                for i = 1:step
                    if 1 <= x && x <= N && 1 <= y && y <= M
                        drawnLines(x, y) = 1;
                    end
                end
            end
        end
    end
end

```

```

        drawnLines(round(y), round(x)) = 1;
    end

    x = x + incX;
    y = y + incY;
end
else
    % Draw a vertical line
    x = rho / c;

    if 1 <= x && x <= N
        for y = 1:M
            drawnLines(y, round(x)) = 1;
        end
    end
end
end
end

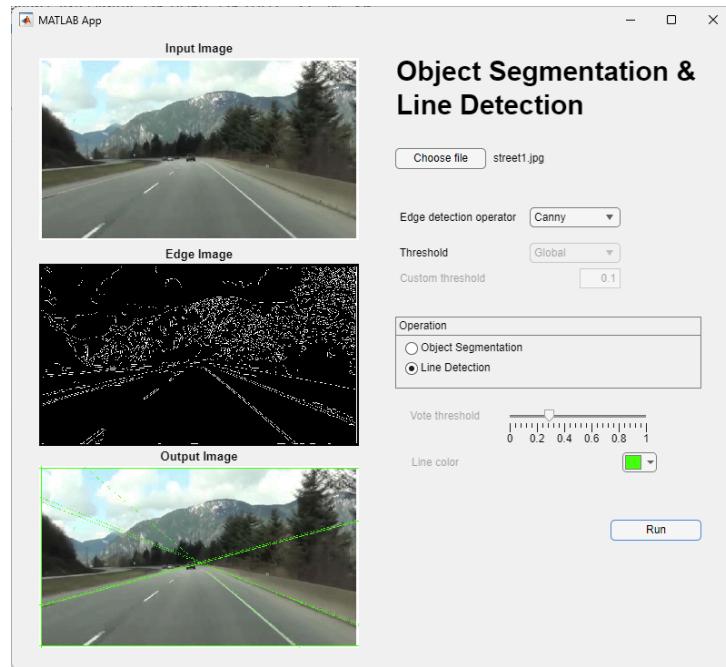
```

Fungsi `lineDraw` menerima empat parameter, yaitu `lines` (matriks hasil pendektsian garis lurus), `M` dan `N` (ukuran citra luaran), serta `threshold` (nilai ambang batas suara). Fungsi ini merupakan gabungan dari pengambangan dan transformasi Hough balikan, dan mengembalikan sebuah citra luaran dengan ukuran $M \times N$ yang memuat semua garis yang ditemukan.

Untuk setiap elemen dari matriks `lines` yang memiliki suara setidaknya sebanyak `threshold`, fungsi ini menggambarkan sebuah garis lurus dalam citra luaran menggunakan parameter (ρ, θ) yang bersesuaian dengan elemen tersebut. Penggambaran garis dilakukan menggunakan algoritma DDA (*digital differential analyzer*).

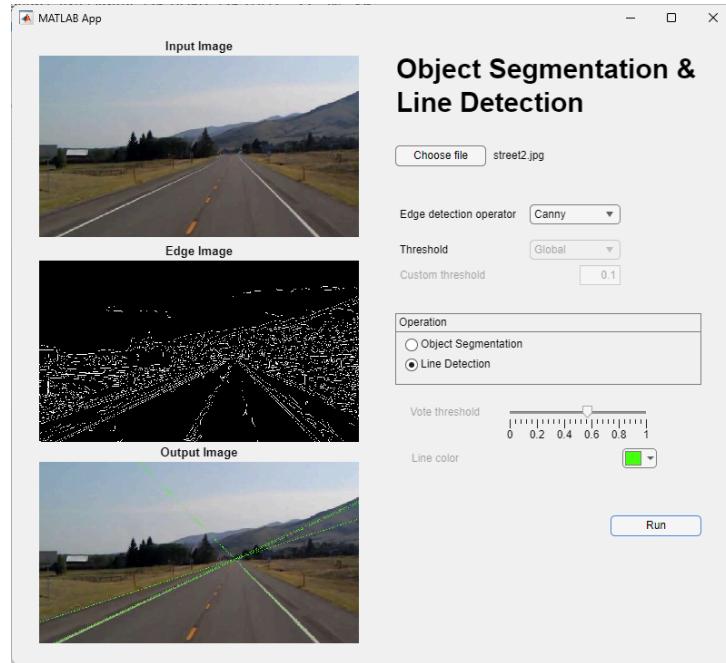
Berikut merupakan contoh hasil eksekusi program.

Deteksi jalur garis jalan raya 1 (*street1.jpg*):



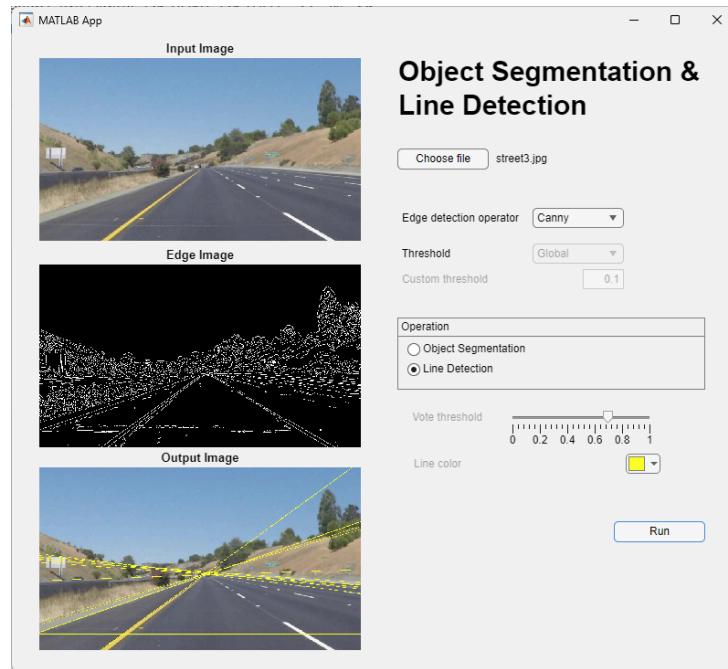
Gambar 2.51 Deteksi jalur garis jalan raya 1

Deteksi jalur garis jalan raya 2 (*street2.jpg*) dengan operator Canny:



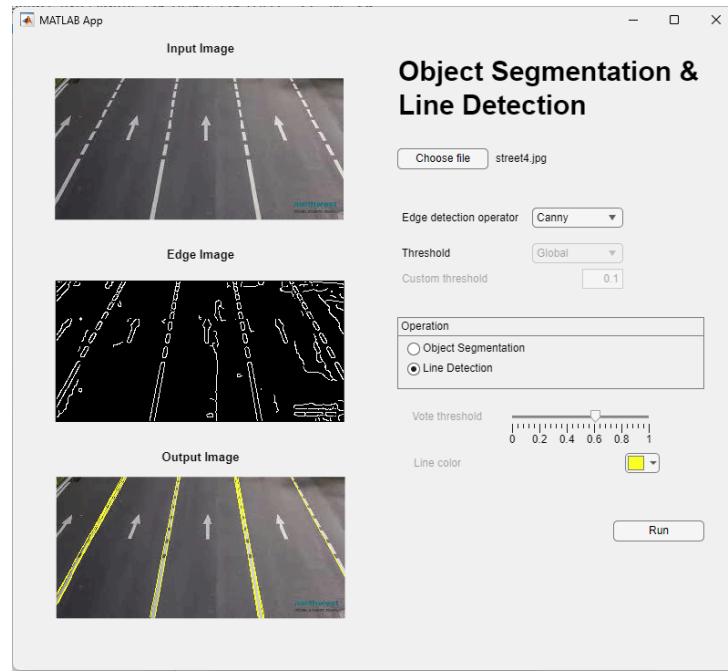
Gambar 2.52 Deteksi jalur garis jalan raya 2

Deteksi jalur garis jalan raya 3 (*street3.jpg*):



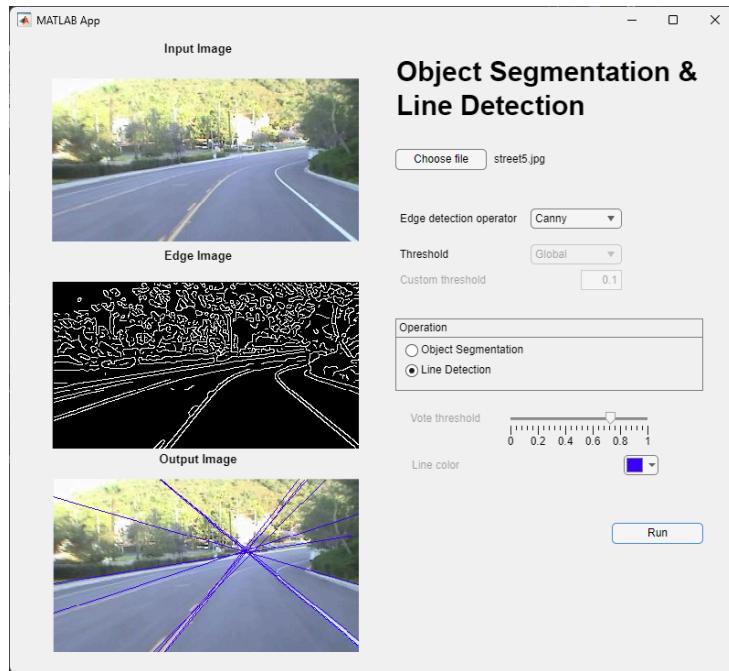
Gambar 2.53 Deteksi jalur garis jalan raya 3

Deteksi jalur garis jalan raya 4 (*street4.jpg*):



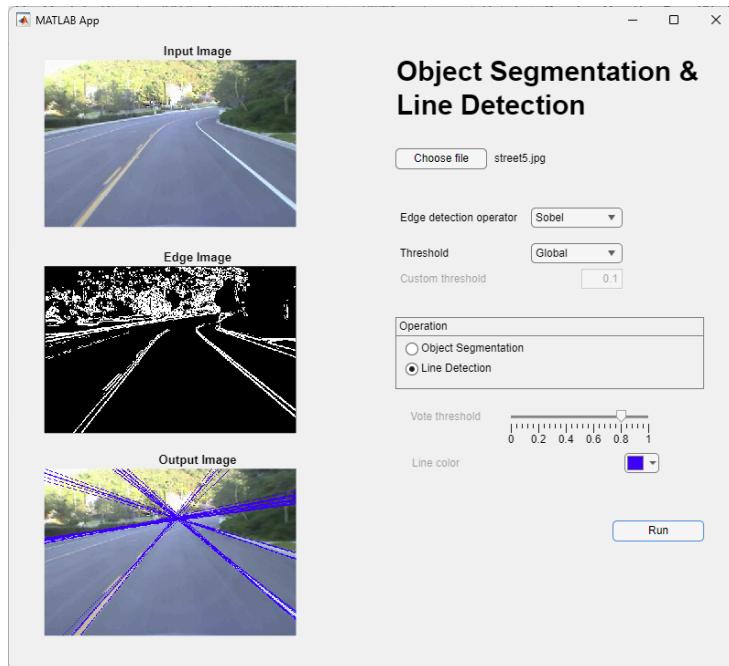
Gambar 2.54 Deteksi jalur garis jalan raya 4

Deteksi jalur garis jalan raya 5 (*street5.jpg*) dengan operator Canny:



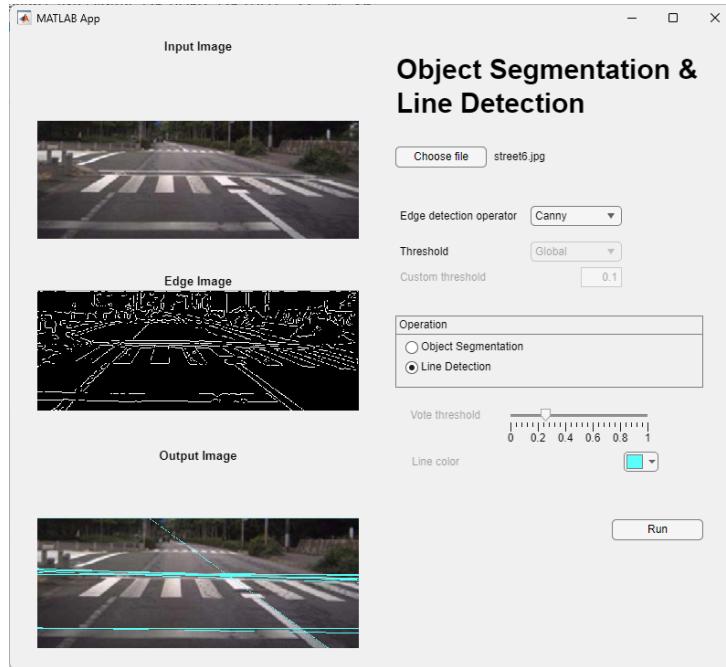
Gambar 2.55 Deteksi jalur garis jalan raya 5 dengan operator Canny

Deteksi jalur garis jalan raya 5 (*street5.jpg*) dengan operator Sobel:



Gambar 2.56 Deteksi jalur garis jalan raya 5 dengan operator Sobel

Deteksi jalur garis jalan raya 6 (*street6.jpg*):



Gambar 2.57 Deteksi jalur garis jalan raya 6

Analisis hasil:

Berdasarkan hasil deteksi garis terhadap citra-citra uji di atas, program mampu menemukan garis-garis lurus yang membatasi jalur jalan raya dalam citra. Namun, hasil tersebut cukup bergantung kepada nilai ambang yang digunakan untuk memilih garis hasil. Pada program ini, nilai ambang relatif terhadap maksimum global dari matriks hasil transformasi Hough.

Algoritma deteksi garis yang diterapkan bekerja dengan baik pada citra uji 1 dan 6 hanya dengan nilai ambang yang relatif rendah akibat adanya *margin* pada citra, yang menyebabkan banyaknya garis yang terdeteksi secara keliru pada daerah pinggir citra.

Citra uji 5 digunakan untuk membandingkan hasil deteksi garis dengan operator deteksi tepi yang berbeda. Secara khusus, deteksi garis dilakukan dengan citra tepi yang dihasilkan dengan metode Sobel dan Canny. Hasil pengujian menunjukkan bahwa deteksi garis dengan operator Canny menghasilkan garis-garis yang jarang, sedangkan deteksi garis dengan operator Sobel menghasilkan garis-garis yang lebih banyak dan terpadatkan pada daerah-daerah tepi. Hasil ini disebabkan karena daerah tepi hasil deteksi dengan operator Canny hanya setebal satu piksel, sedangkan operator Sobel menghasilkan daerah tepi yang lebih tebal. Semakin tebal

sebuah daerah tepi dihasilkan dari sebuah operator deteksi tepi, semakin lebar puncak yang bersesuaian dengan daerah tersebut pada matriks hasil transformasi Hough.

3. Lampiran

Berikut merupakan tautan repositori *GitHub*:

https://github.com/arleenchr/IF4073_Tugas3