

Tugas 3 IF4073 Interpretasi dan Pengolahan Citra



Disusun oleh:

13520065 Rayhan Kinan Muhamnad
13520123 Johannes Winson Sukiatmodjo

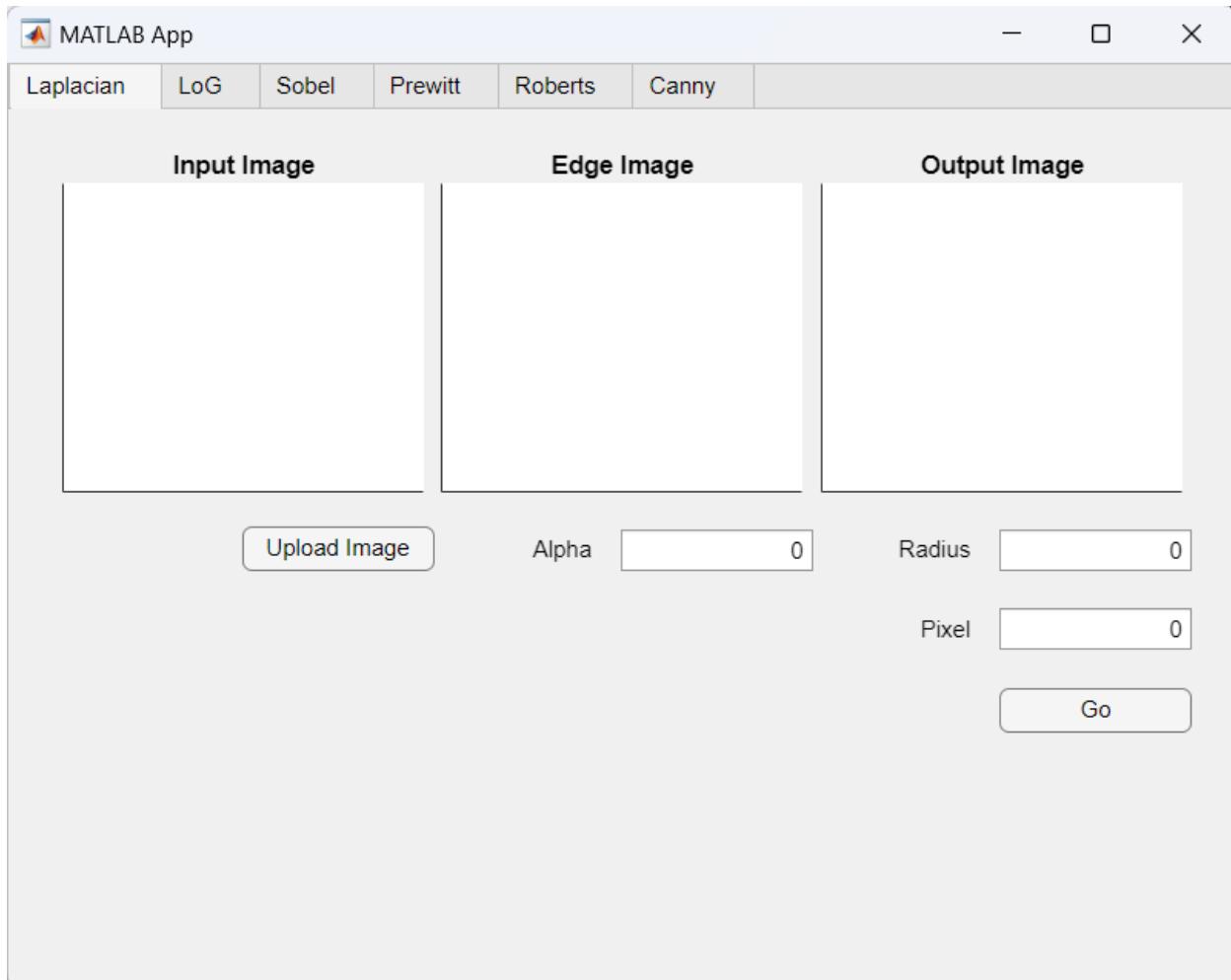
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2023

Screenshot GUI program

Berikut disajikan laman GUI program yang kami buat secara umum. Dalam GUI program ini, terdapat:

1. *Tab* untuk navigasi dalam aplikasi untuk memilih jenis operator deteksi tepi yang ingin digunakan.
2. Panel utama yang berisikan *input image*, *edge image*, *output image*, serta tombol-tombol untuk mengoperasikannya.

Untuk lebih detailnya, GUI untuk setiap fitur yang kami buat akan kami lampirkan pada bagian selanjutnya pada masing-masing fitur.



Operator 1: Laplace

Pada bagian ini, kami akan menjelaskan detail operator pertama kami yaitu operator Laplace.

Kode program

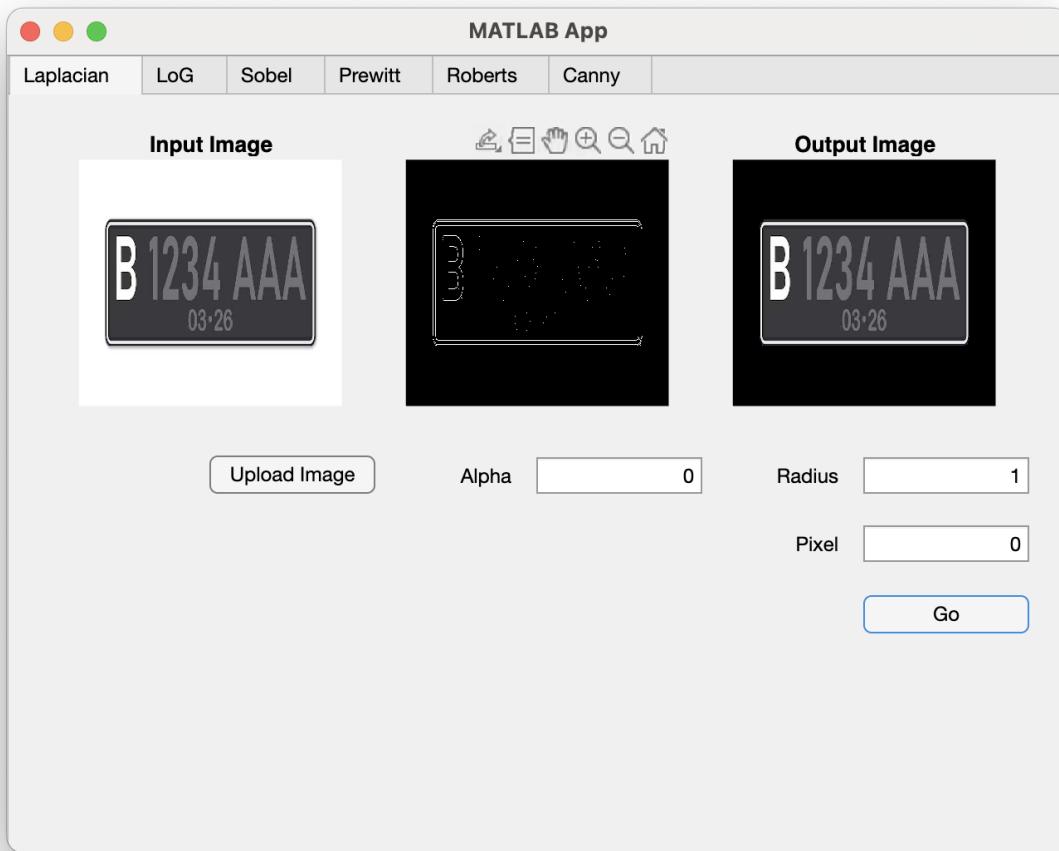
Berikut adalah fungsi operator Laplace yang kami buat:

```
function paddedData = ApplyLaplacian(imageData, alpha)
    % arguments
    % imageData uint8
    % alpha double {mustBeGreaterThanOrEqual(alpha, 0),
    % mustBeLessThanOrEqual(alpha, 1)}
    % end

    % Calculate Laplacian
    doubleImageData = im2double(imageData);
    h = fspecial('laplacian', alpha);
    doubleResultData = conv2(doubleImageData, h, 'valid');
    resultData = im2uint8(doubleResultData);
    binarizeData = imbinarize(resultData);

    % Pad result
    [imageHeight, imageWidth] = size(imageData);
    [hHeight, hWidth] = size(h);
    padHeight = floor(hHeight / 2);
    padWidth = floor(hWidth / 2);
    paddedData = zeros(imageHeight, imageWidth);
    paddedData(padHeight + 1:imageHeight - padHeight, padWidth + 1:imageWidth - padWidth) = binarizeData;
end
```

Contoh hasil eksekusi program



Analisis cara kerja algoritma

Fungsi ApplyLaplacian memiliki tujuan untuk menerapkan filter Laplacian pada citra masukan dengan parameter kontrol tingkat intensitas (alpha) dan menghasilkan citra yang telah diproses dengan efek Laplacian. Fungsi ini dimulai dengan melakukan validasi argumen untuk memastikan bahwa citra masukan (imageData) adalah matriks uint8 dan parameter alpha berada dalam rentang yang valid antara 0 dan 1.

Selanjutnya, citra masukan dikonversi ke tipe data ganda (double), karena filter Laplacian bekerja pada tipe data ganda. Filter Laplacian itu sendiri dihitung menggunakan fspecial dengan parameter alpha, dan kemudian diterapkan pada citra masukan menggunakan operasi konvolusi. Hasilnya dikonversi kembali ke tipe data uint8 agar tetap dalam format citra 8-bit.

Setelah itu, hasil konvolusi dibinarisasi menjadi citra biner menggunakan imbinarize. Akhirnya, untuk mengembalikan citra hasil ke ukuran asli, citra biner dipad dengan nol di sekitar tepi citra sehingga memiliki ukuran yang sama dengan citra asli dengan efek Laplacian yang diterapkan. Hasil akhir adalah citra paddedData yang mencerminkan citra asli yang telah diproses dengan filter Laplacian dan dibinarisasi.

Operator 2: LoG

Pada bagian ini, kami akan menjelaskan detail operator kedua kami yaitu operator LoG.

Kode program

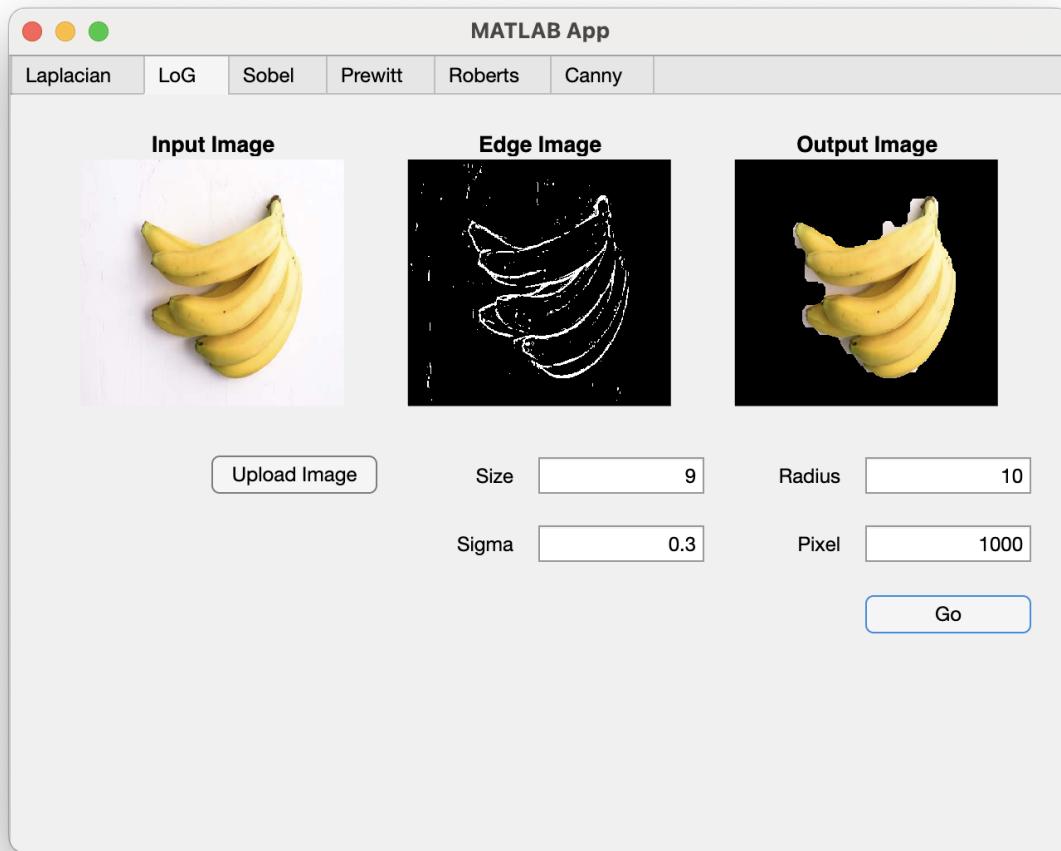
Berikut adalah fungsi operator LoG yang kami buat:

```
function paddedData = ApplyLaplacianOfGaussian(imageData, hsize, sigma)
    arguments
        imageData uint8
        hsize double {mustBePositive, mustBeInteger}
        sigma double {mustBePositive}
    end

    % Calculate Laplacian of Gaussian
    doubleImageData = im2double(imageData);
    h = fspecial('log', hsize, sigma);
    doubleResultData = conv2(doubleImageData, h, 'valid');
    resultData = im2uint8(doubleResultData);
    binarizeData = imbinarize(resultData);

    % Pad result
    [imageHeight, imageWidth] = size(imageData);
    [hHeight, hWidth] = size(h);
    padHeight = floor(hHeight / 2);
    padWidth = floor(hWidth / 2);
    paddedData = zeros(imageHeight, imageWidth);
    paddedData(padHeight + 1:imageHeight - padHeight, padWidth + 1:imageWidth - padWidth) = binarizeData;
end
```

Contoh hasil eksekusi program



Analisis cara kerja algoritma

Fungsi `ApplyLaplacianOfGaussian` bertujuan untuk menerapkan filter Laplacian of Gaussian (LoG) pada citra masukan (`imageData`) dengan parameter ukuran filter (`hsize`) dan nilai sigma (`sigma`) yang mengontrol sifat filter LoG. Tujuannya adalah menghasilkan citra yang telah diproses dengan efek Laplacian of Gaussian. Fungsi ini memulai dengan validasi argumen untuk memastikan bahwa citra masukan adalah matriks `uint8`, ukuran filter adalah bilangan bulat positif, dan nilai sigma adalah bilangan positif.

Selanjutnya, citra masukan dikonversi menjadi tipe data ganda (`double`) menggunakan `im2double`, karena filter LoG bekerja pada tipe data ganda. Filter LoG itu sendiri dihitung menggunakan `fspecial` dengan parameter `hsize` dan `sigma`, dan kemudian diterapkan pada citra masukan menggunakan operasi konvolusi. Hasilnya dikonversi kembali ke tipe data `uint8` agar tetap dalam format citra 8-bit.

Setelah itu, hasil konvolusi dibinarisasi menjadi citra biner menggunakan imbinarize. Akhirnya, untuk mengembalikan citra hasil ke ukuran asli, citra biner dipad dengan nol di sekitar tepi citra sehingga memiliki ukuran yang sama dengan citra asli dengan efek LoG yang diterapkan. Hasil akhir adalah citra paddedData, yang mencerminkan citra asli yang telah diproses dengan filter Laplacian of Gaussian dan dibinarisasi.

Operator 3: Sobel

Pada bagian ini, kami akan menjelaskan detail operator ketiga kami yaitu operator Sobel.

Kode program

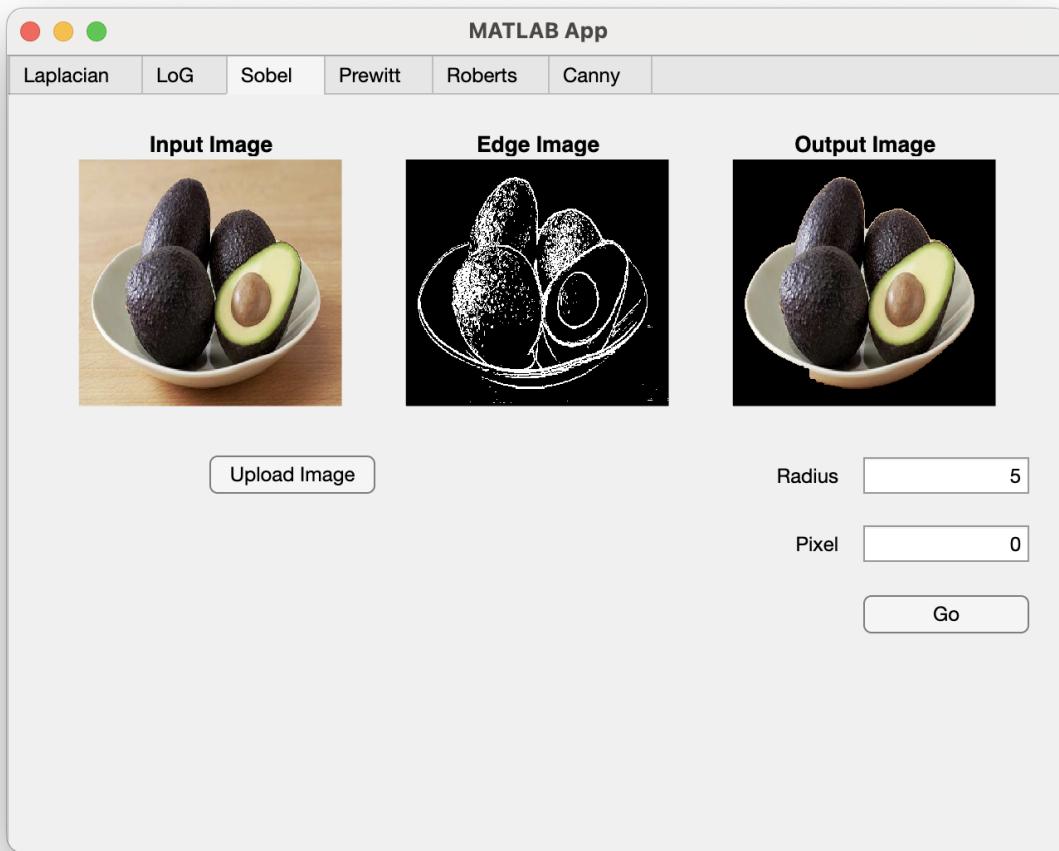
Berikut adalah fungsi operator Sobel yang kami buat:

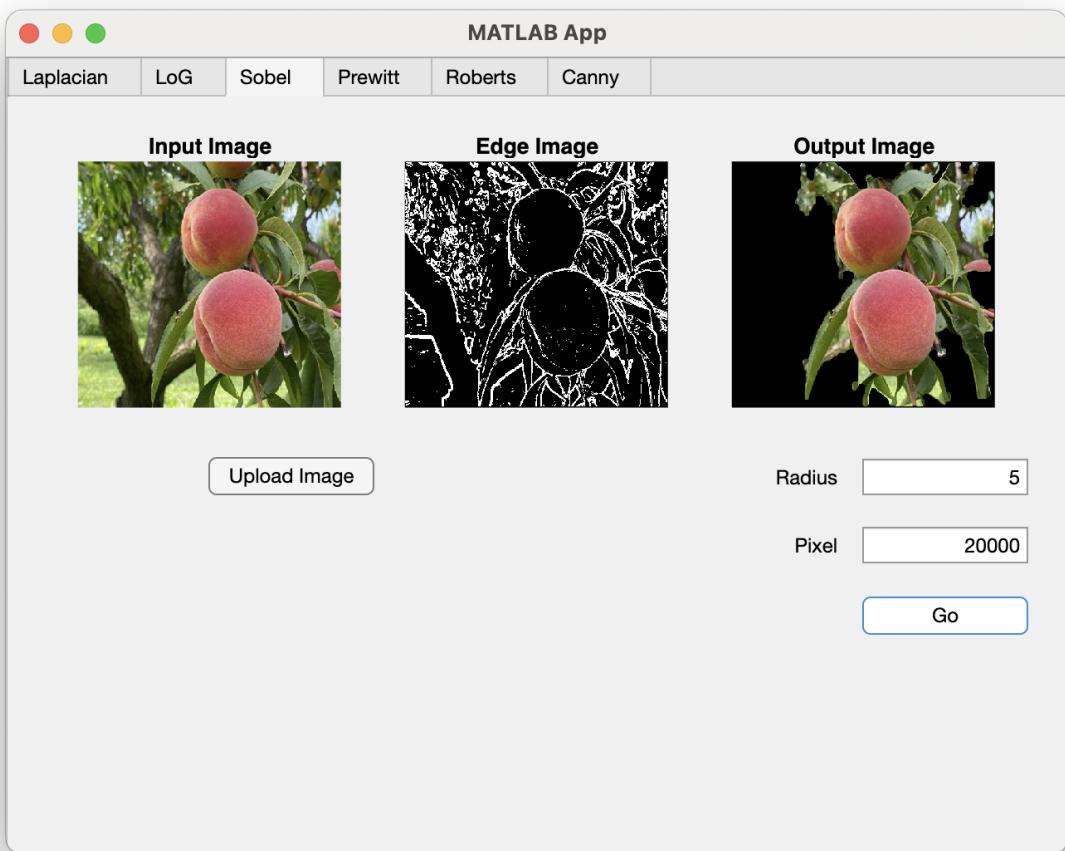
```
function paddedData = ApplySobel(imageData)
    arguments
        imageData uint8
    end

    % Calculate Sobel
    doubleImageData = im2double(imageData);
    h = fspecial('sobel');
    v = h';
    doubleResultData = sqrt(conv2(doubleImageData, h, 'valid') .^ 2 +
conv2(doubleImageData, v, 'valid') .^ 2);
    resultData = im2uint8(doubleResultData);
    binarizeData = imbinarize(resultData);

    % Pad result
    [imageHeight, imageWidth] = size(imageData);
    [hHeight, hWidth] = size(h);
    padHeight = floor(hHeight / 2);
    padWidth = floor(hWidth / 2);
    paddedData = zeros(imageHeight, imageWidth);
    paddedData(padHeight + 1:imageHeight - padHeight, padWidth + 1:imageWidth -
padWidth) = binarizeData;
end
```

Contoh hasil eksekusi program





Analisis cara kerja algoritma

Fungsi `ApplySobel` bertujuan untuk menerapkan operator Sobel pada citra masukan (`imageData`) dengan tujuan mendeteksi tepi dalam citra. Fungsi ini dimulai dengan validasi argumen untuk memastikan bahwa citra masukan adalah matriks `uint8`, yang merupakan format citra dalam 8-bit unsigned integer.

Selanjutnya, citra masukan dikonversi menjadi tipe data ganda (double) menggunakan `im2double`, karena operator Sobel bekerja dengan tipe data ganda. Operator Sobel itu sendiri dihitung untuk mengukur gradien dalam arah horizontal dan vertikal. Kemudian, operator Sobel tersebut diterapkan pada citra masukan dengan operasi konvolusi, menghasilkan gradien citra dalam kedua arah.

Gradien dalam arah horizontal dan vertikal kemudian digabungkan dengan menghitung gradien magnitude citra menggunakan rumus Pythagoras (akar kuadrat dari jumlah kuadrat gradien

dalam kedua arah). Hasil gradien magnitude dikonversi kembali ke tipe data uint8 agar tetap dalam format citra 8-bit.

Selanjutnya, hasil gradien magnitude dibinarisasi dengan imbinarize, menghasilkan citra biner yang menunjukkan lokasi tepi dalam citra. Akhirnya, citra biner tersebut dipad dengan nol di sekitar tepi citra asli sehingga memiliki ukuran yang sama dengan citra asli. Hasil akhir adalah citra paddedData, yang mencerminkan citra asli yang telah diproses dengan operator Sobel untuk deteksi tepi dan diubah menjadi citra biner.

Operator 4: Prewitt

Pada bagian ini, kami akan menjelaskan detail operator keempat kami yaitu operator Prewitt.

Kode program

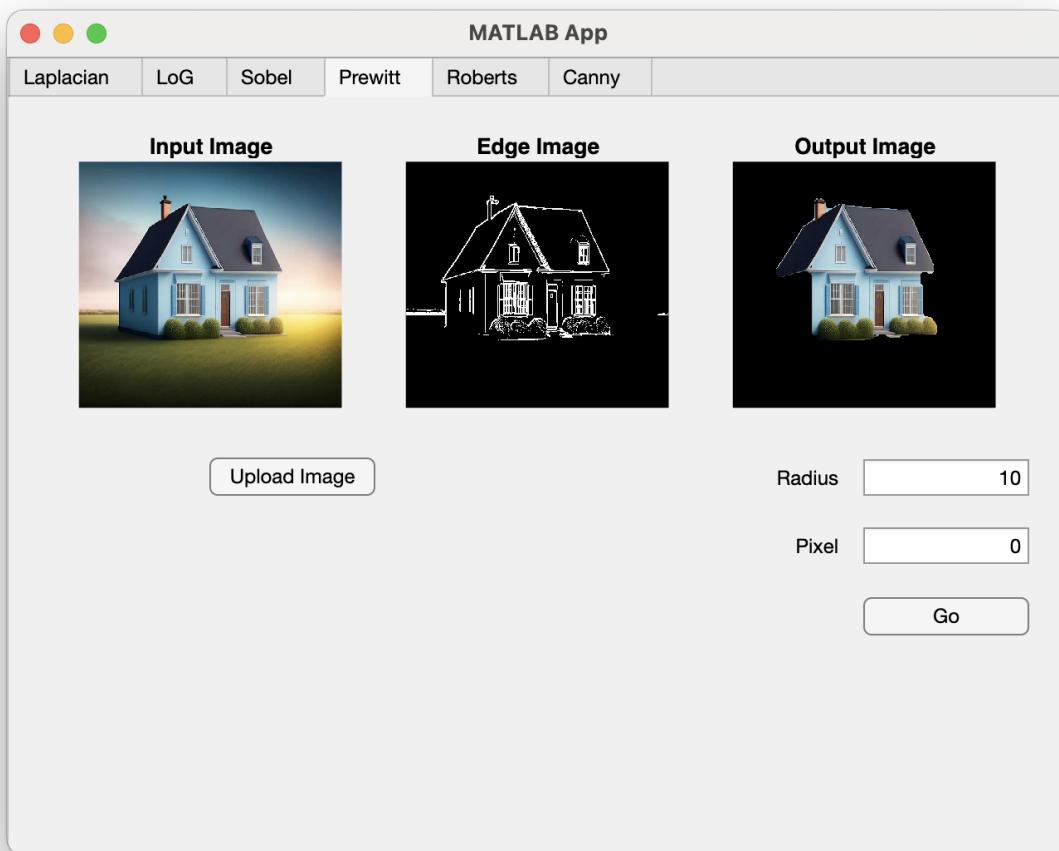
Berikut adalah fungsi operator Prewitt yang kami buat:

```
function paddedData = ApplyPrewitt(imageData)
    arguments
        imageData uint8
    end

    % Calculate Prewitt
    doubleImageData = im2double(imageData);
    h = fspecial('prewitt');
    v = h';
    doubleResultData = sqrt(conv2(doubleImageData, h, 'valid') .^ 2 +
conv2(doubleImageData, v, 'valid') .^ 2);
    resultData = im2uint8(doubleResultData);
    binarizeData = imbinarize(resultData);

    % Pad result
    [imageHeight, imageWidth] = size(imageData);
    [hHeight, hWidth] = size(h);
    padHeight = floor(hHeight / 2);
    padWidth = floor(hWidth / 2);
    paddedData = zeros(imageHeight, imageWidth);
    paddedData(padHeight + 1:imageHeight - padHeight, padWidth + 1:imageWidth -
padWidth) = binarizeData;
end
```

Contoh hasil eksekusi program



Analisis cara kerja algoritma

Fungsi `ApplyPrewitt` bertujuan untuk menerapkan operator Prewitt pada citra masukan (`imageData`) dengan tujuan mendeteksi tepi dalam citra. Fungsi ini dimulai dengan validasi argumen untuk memastikan bahwa citra masukan adalah matriks `uint8`, yang merupakan format citra dalam 8-bit unsigned integer.

Selanjutnya, citra masukan dikonversi menjadi tipe data ganda (`double`) menggunakan `im2double`, karena operator Prewitt bekerja dengan tipe data ganda. Operator Prewitt itu sendiri dihitung untuk mengukur gradien dalam arah horizontal dan vertikal. Kemudian, operator Prewitt tersebut diterapkan pada citra masukan dengan operasi konvolusi, menghasilkan gradien citra dalam kedua arah.

Gradien dalam arah horizontal dan vertikal kemudian digabungkan dengan menghitung gradien magnitude citra menggunakan rumus Pythagoras (akar kuadrat dari jumlah kuadrat gradien dalam kedua arah). Hasil gradien magnitude dikonversi kembali ke tipe data uint8 agar tetap dalam format citra 8-bit.

Selanjutnya, hasil gradien magnitude dibinarisasi dengan imbinarize, menghasilkan citra biner yang menunjukkan lokasi tepi dalam citra. Akhirnya, citra biner tersebut dipad dengan nol di sekitar tepi citra asli sehingga memiliki ukuran yang sama dengan citra asli. Hasil akhir adalah citra paddedData, yang mencerminkan citra asli yang telah diproses dengan operator Prewitt untuk deteksi tepi dan diubah menjadi citra biner.

Operator 5: Roberts

Pada bagian ini, kami akan menjelaskan detail operator kelima kami yaitu operator Roberts.

Kode program

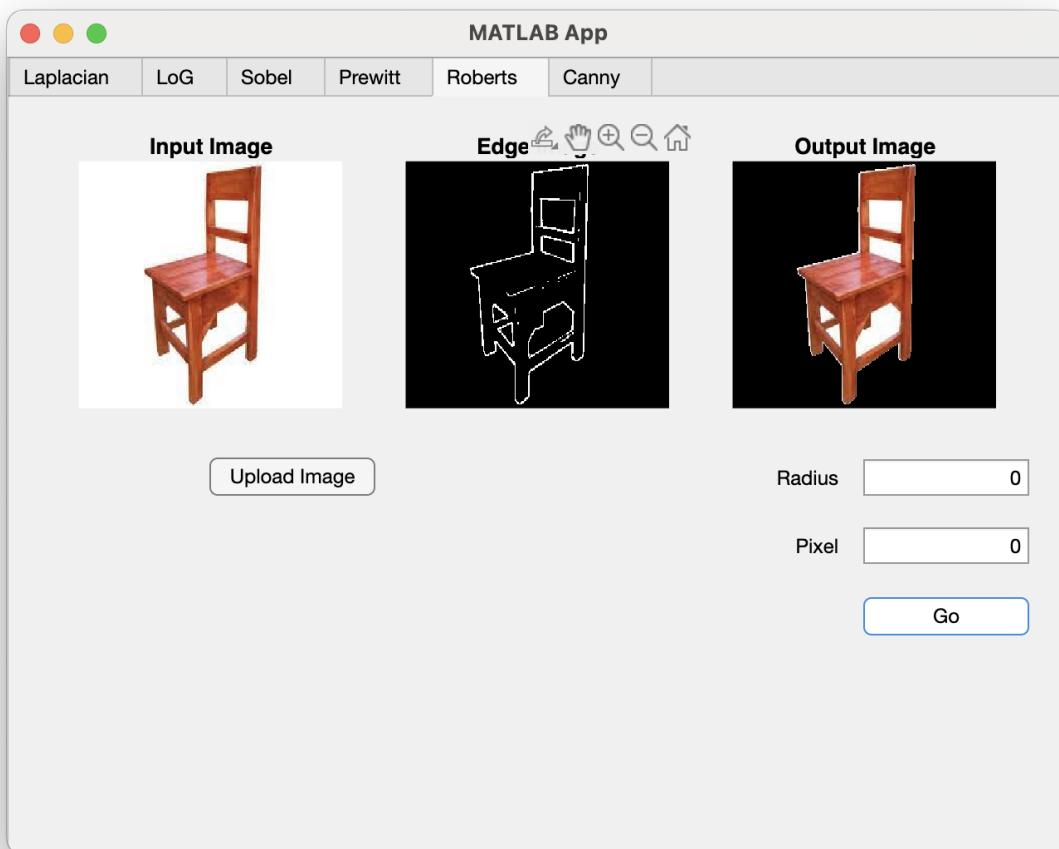
Berikut adalah fungsi operator Roberts yang kami buat:

```
function paddedData = ApplyRoberts(imageData)
    arguments
        imageData uint8
    end

    % Calculate Roberts
    doubleImageData = im2double(imageData);
    rPlus = [1 0; 0 -1];
    rMinus = [0 1; -1 0];
    doubleResultData = sqrt(conv2(doubleImageData, rPlus, 'valid') .^ 2 +
conv2(doubleImageData, rMinus, 'valid') .^ 2);
    resultData = im2uint8(doubleResultData);
    binarizeData = imbinarize(resultData);

    % Pad result
    [imageHeight, imageWidth] = size(imageData);
    [hHeight, hWidth] = size(rPlus);
    padHeight = floor(hHeight / 2);
    padWidth = floor(hWidth / 2);
    paddedData = zeros(imageHeight, imageWidth);
    paddedData(padHeight:imageHeight - padHeight, padWidth:imageWidth -
padWidth) = binarizeData;
end
```

Contoh hasil eksekusi program



Analisis cara kerja algoritma

Fungsi ApplyRoberts bertujuan untuk menerapkan operator Roberts pada citra masukan (imageData) dengan tujuan mendeteksi tepi dalam citra. Fungsi ini memulai dengan validasi argumen untuk memastikan bahwa citra masukan adalah matriks uint8, yang merupakan format citra dalam 8-bit unsigned integer.

Selanjutnya, citra masukan dikonversi menjadi tipe data ganda (double) menggunakan im2double, karena operator Roberts bekerja dengan tipe data ganda. Operator Roberts terdiri dari dua kernel, yaitu rPlus dan rMinus, yang digunakan untuk menghitung gradien dalam arah diagonal positif dan negatif. Kemudian, kedua kernel ini diterapkan pada citra masukan dengan operasi konvolusi, menghasilkan gradien citra dalam kedua arah diagonal.

Gradien dalam arah diagonal yang positif dan negatif digabungkan dengan menghitung gradien magnitude citra menggunakan rumus Pythagoras (akar kuadrat dari jumlah kuadrat gradien dalam kedua arah diagonal). Hasil gradien magnitude dikonversi kembali ke tipe data uint8 agar tetap dalam format citra 8-bit.

Selanjutnya, hasil gradien magnitude dibinarisasi dengan imbinarize, menghasilkan citra biner yang menunjukkan lokasi tepi dalam citra. Akhirnya, citra biner tersebut dipad dengan nol di sekitar tepi citra asli sehingga memiliki ukuran yang sama dengan citra asli. Hasil akhir adalah citra paddedData, yang mencerminkan citra asli yang telah diproses dengan operator Roberts untuk deteksi tepi dengan menggunakan arah diagonal, dan diubah menjadi citra biner.

Operator 6: Canny

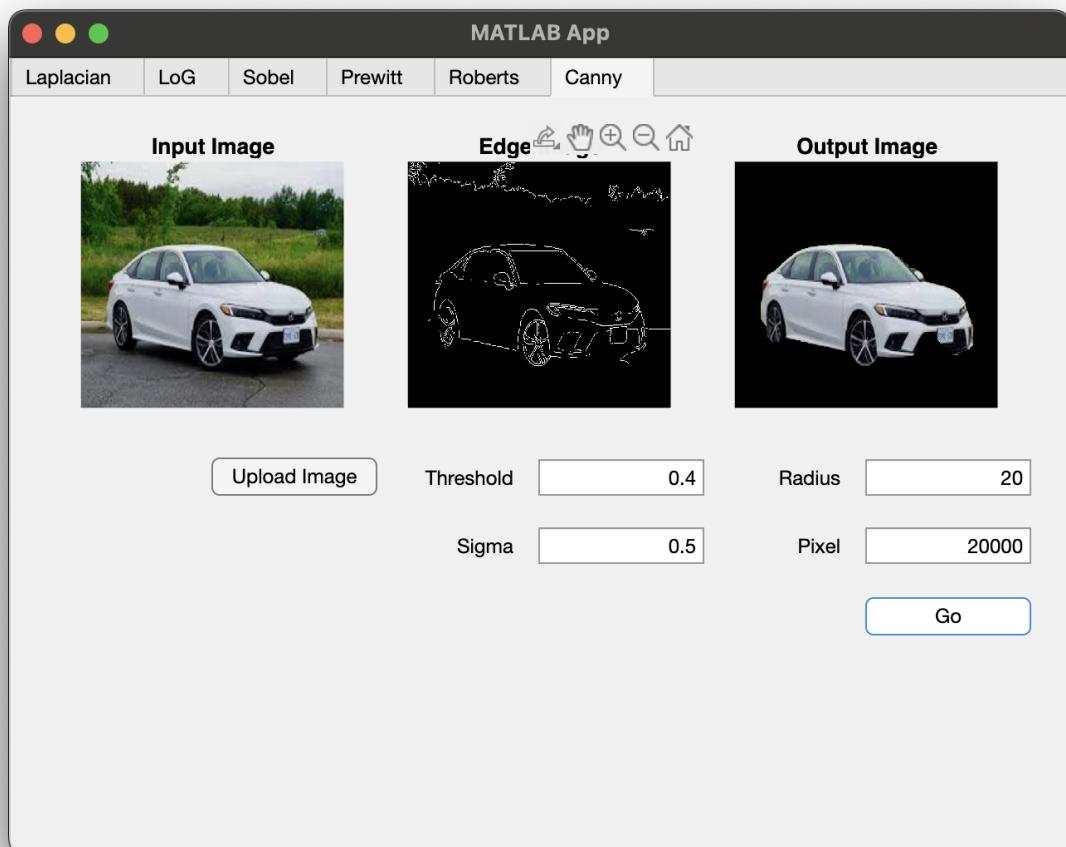
Pada bagian ini, kami akan menjelaskan detail operator keenam kami yaitu operator Canny.

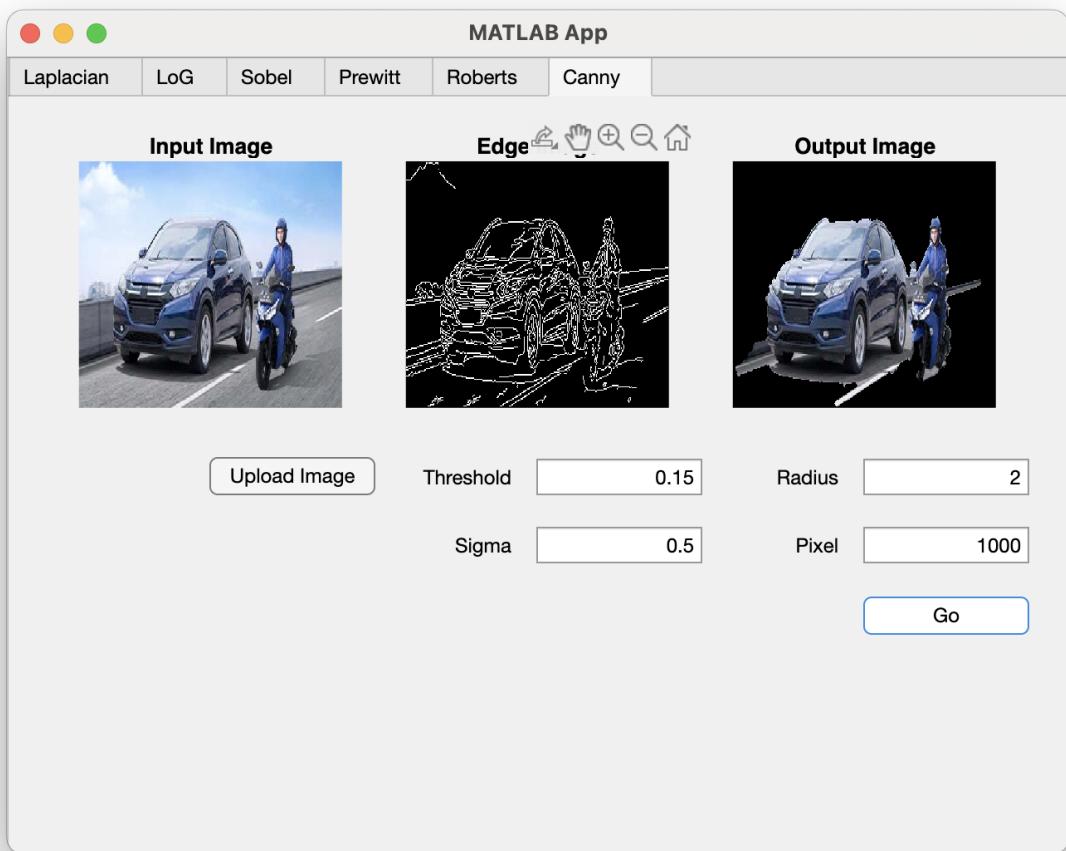
Kode program

Berikut adalah operator Canny yang kami buat:

```
 imageData = edge(doubleImageData, 'Canny', threshold, sigma);
```

Contoh hasil eksekusi program





Analisis cara kerja algoritma

Algoritma di atas merupakan pemanggilan fungsi edge dalam MATLAB untuk mendeteksi tepi dalam citra dengan menggunakan metode deteksi tepi Canny. Hasil dari pemanggilan fungsi edge ini adalah citra biner (hitam-putih) yang disimpan kembali dalam variabel imageData. Citra tersebut akan memiliki tepi yang ditandai sebagai piksel putih, sedangkan latar belakangnya akan menjadi piksel hitam, sesuai dengan ambang yang telah ditentukan.

Alamat GitHub program

GitHub program dapat diakses pada pranala berikut: [Pranala GitHub](#)