

Laporan Tugas Kecil 3
IF4073 Interpretasi dan Pengolahan Citra



Disusun oleh:

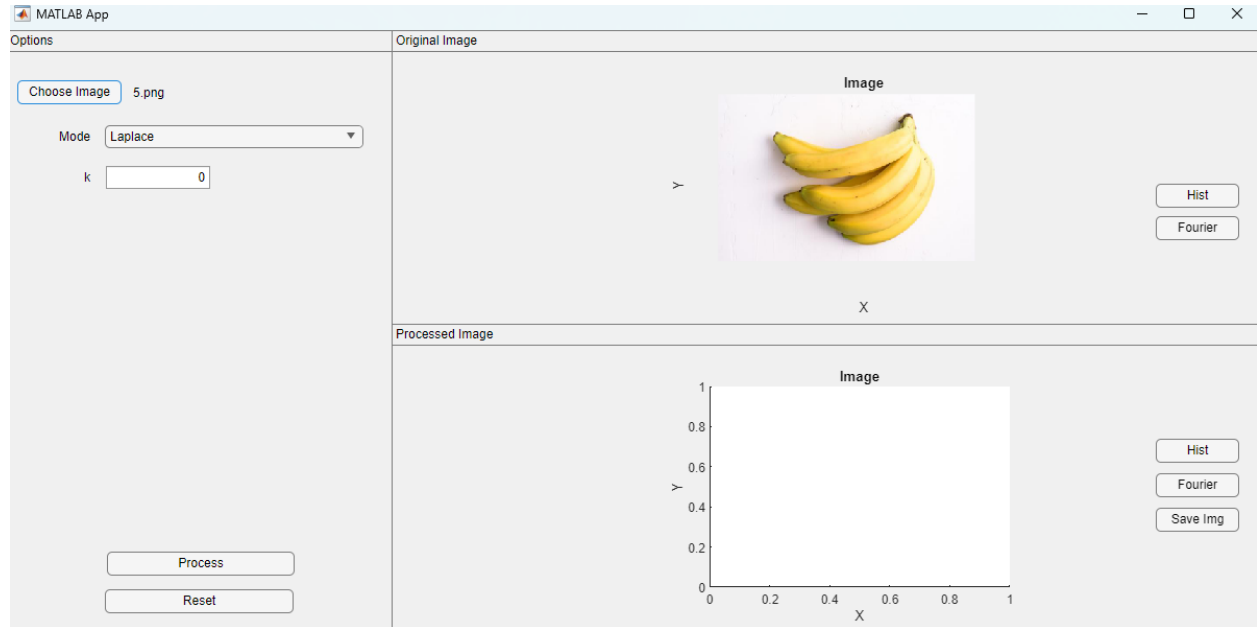
Safiq Faray	13519145
Louis Yanggara	13520063

Tanggal Pengumpulan:

1 November 2023

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023

GUI



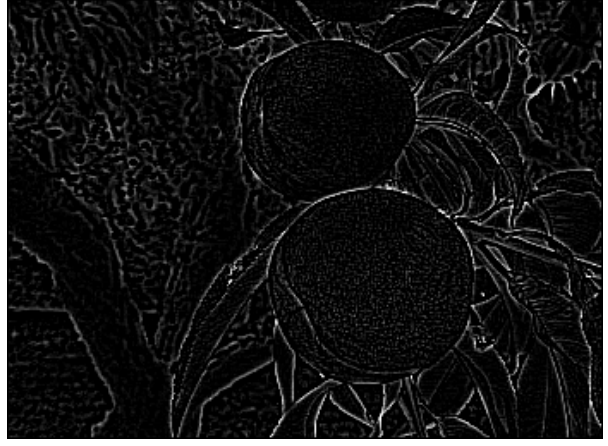
Task 1. Laplace

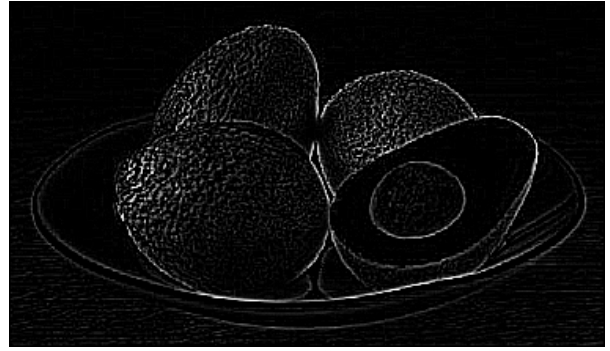
Kode Program

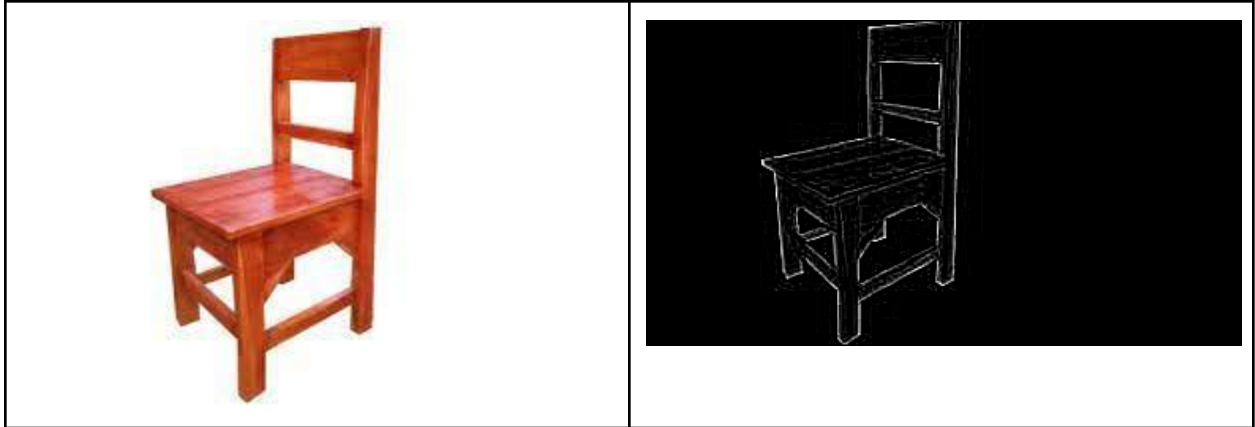
```
function result = laplace(img)
    img = rgb2gray(img);
    R = [1 1 1; 1 -8 1; 1 1 1];
    result = uint8(convn(double(img), double(R), 'same'));
end
```

Hasil Eksekusi

Gambar	Hasil Deteksi Tepi
--------	--------------------







Analisis

Laplace merupakan operator pendeteksi tepi dengan turunan kedua. Operator laplace mendeteksi tepi yang lebih akurat dibandingkan operator turunan pertama, terutama deteksi tepi yang curam. Namun, terkadang operator ini dapat menghasilkan tepi palsu, maka pada hasil eksekusi, dilakukan penghalusan gambar dengan gaussian filter. Untuk mask konvolusi laplace operator yang digunakan adalah $\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$.

Task 2. LoG

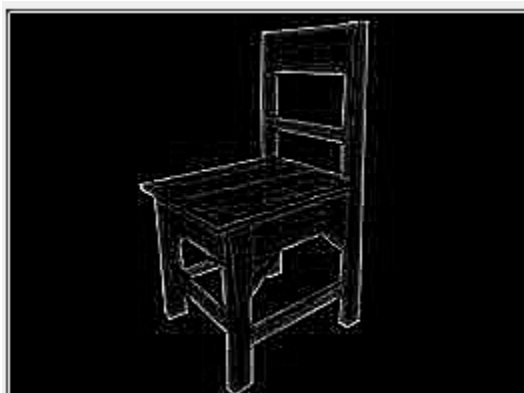
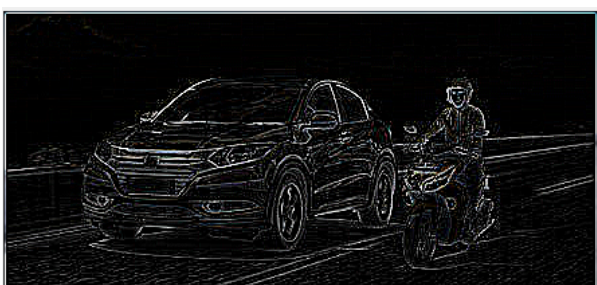
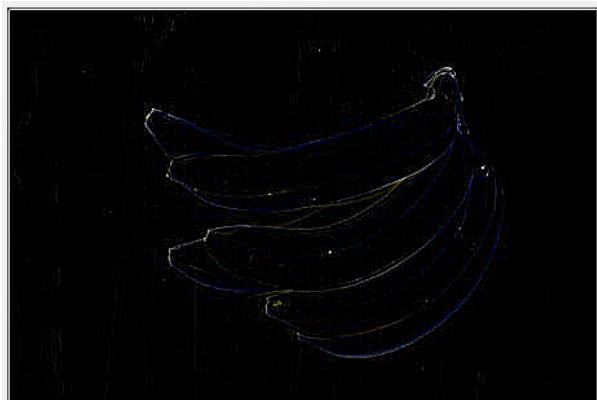
Kode Program

```
function result = log(img)
    h = fspecial("log");
    result = uint8(convn(double(img), double(h)));
    figure, imshow(result);
end
```

Hasil Eksekusi

Gambar	Hasil Deteksi Tepi
--------	--------------------





Analisis



Biasanya LoG digunakan jika gambar memiliki derau yang menyebabkan adanya tepi-tepi palsu, oleh karena itu dilakukan penapisan terlebih dahulu dengan Gaussian kemudian dilakukan operasi Laplace, pada program yang dibuat, fungsi log dibangkitkan dengan menggunakan fspecial sehingga dapat langsung melakukan penapisan sekaligus deteksi tepi.

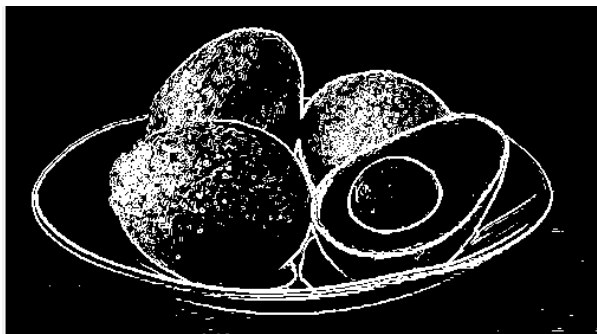
Task 3. Sobel

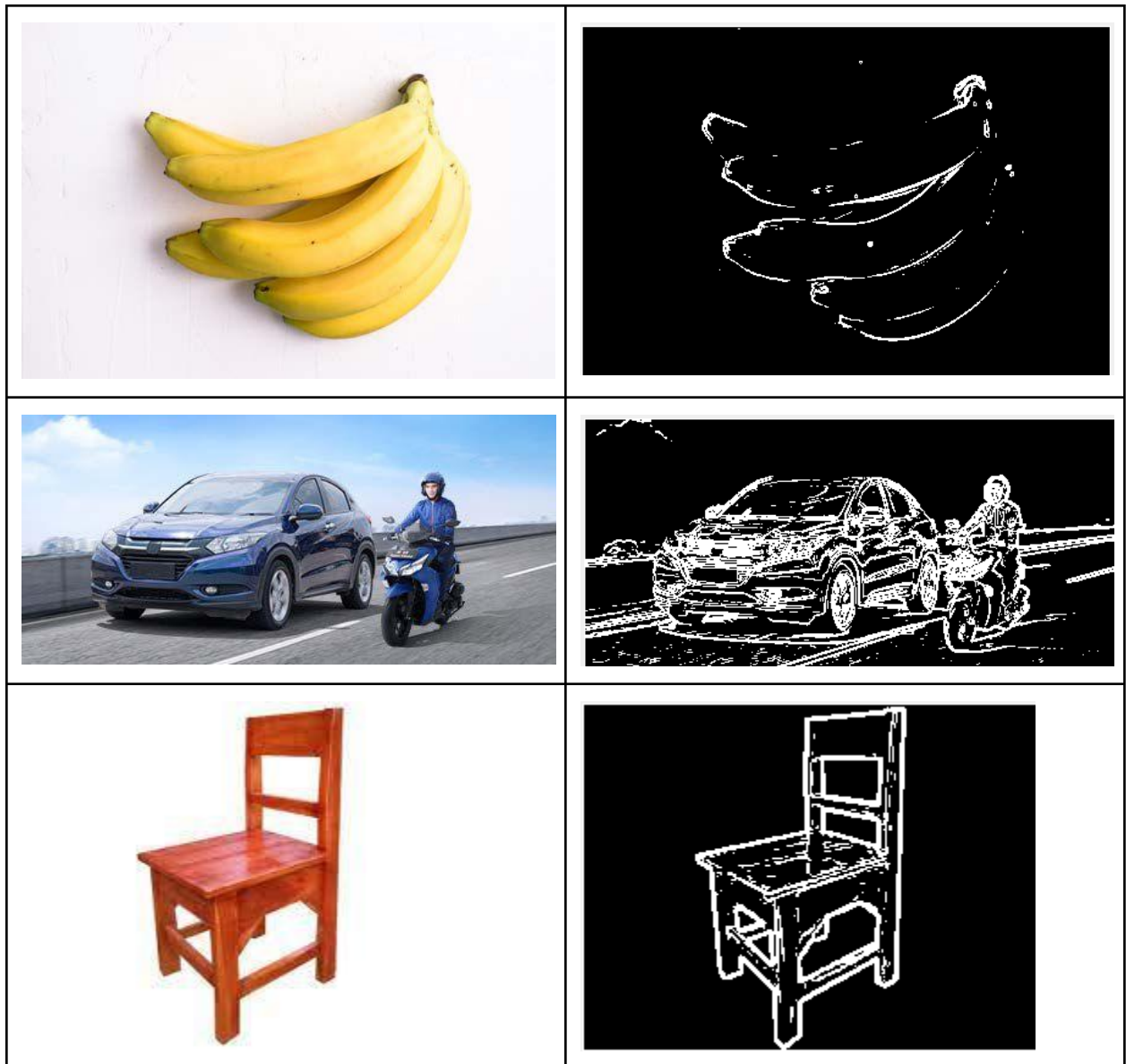
Kode Program

```
function result = sobel(img)
    Sx = [-1 0 1;-2 0 2;-1 0 1];
    Sy = [1 2 1;0 0 0;-1 -2 -1];
    img = rgb2gray(img);
    Jx = convn(double(img), double(Sx), 'same');
    Jy = convn(double(img), double(Sy), 'same');
    result = uint8(sqrt(Jx.^2 + Jy.^2));
    result(result>100) = 255;
    result(result~=255) = 0;
    %figure, imshow(result);
end
```

Hasil Eksekusi

Gambar	Hasil Deteksi Tepi
	





Analisis


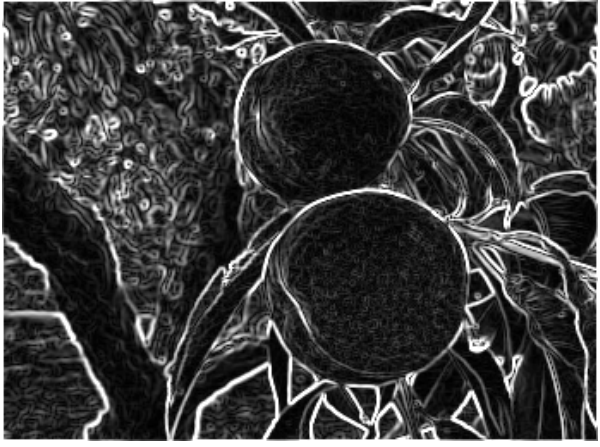


Pada algoritma Sobel, akan digunakan magnitudo dari gradien yang dihitung dengan menggunakan turunan parsial. Untuk mempermudah, digunakan mask S_x dan S_y yang sudah sesuai dengan perhitungan sehingga cukup dilakukan konvolusi dengan menggunakan S_x dan S_y yang hasilnya akan dihitung magnitudonya untuk menentukan nilai akhir pixel. Setelah nilai akhir diperoleh dilakukan pengambangan dengan menggunakan treshold 100,

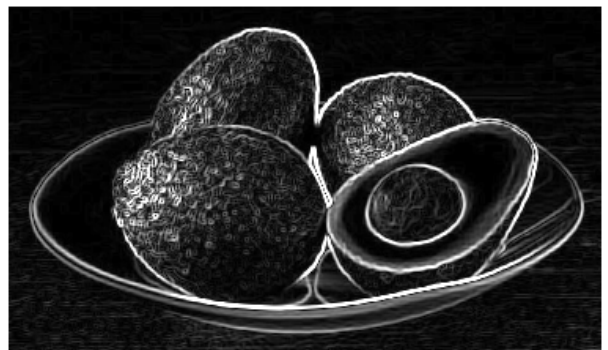
Task 4. Prewitt

Kode Program

```
function result = prewitt(img)
    img = rgb2gray(img);
    Rx = [-1 0 1;-1 0 1; -1 0 1];
    Ry = [1 1 1;0 0 0; -1 -1 -1];
    Jx = convn(double(img), double(Rx), 'same');
    Jy = convn(double(img), double(Ry), 'same');
    result = uint8(sqrt(Jx.^2 + Jy.^2));
end
```

Hasil Eksekusi

Gambar	Hasil Deteksi Tepi
	
	





Analisis


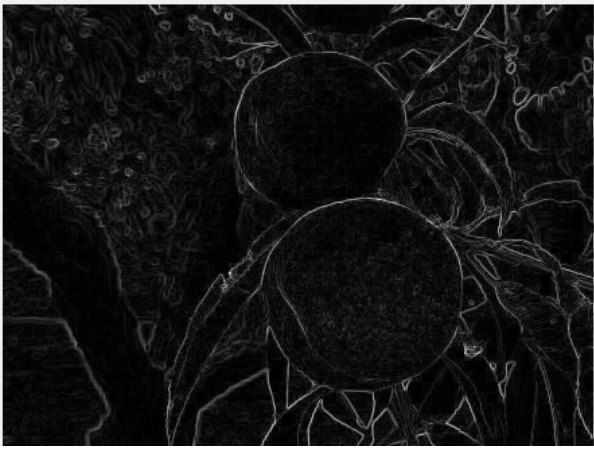




Memiliki turunan parsial yang sama seperti sobel, namun dengan nilai $c=1$. Operator ini digunakan untuk mendeteksi garis vertikal dan horizontal pada gambar, sehingga gambar dengan lengkungan seperti pisang, tepinya tidak terdeteksi dengan baik.

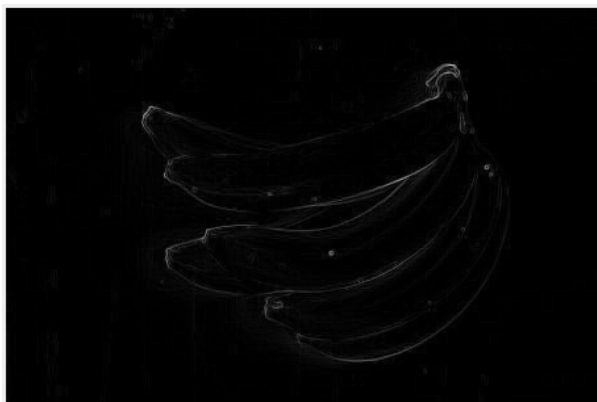
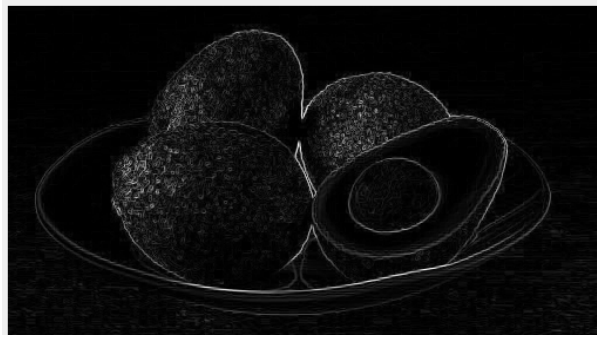
Task 5. Robert

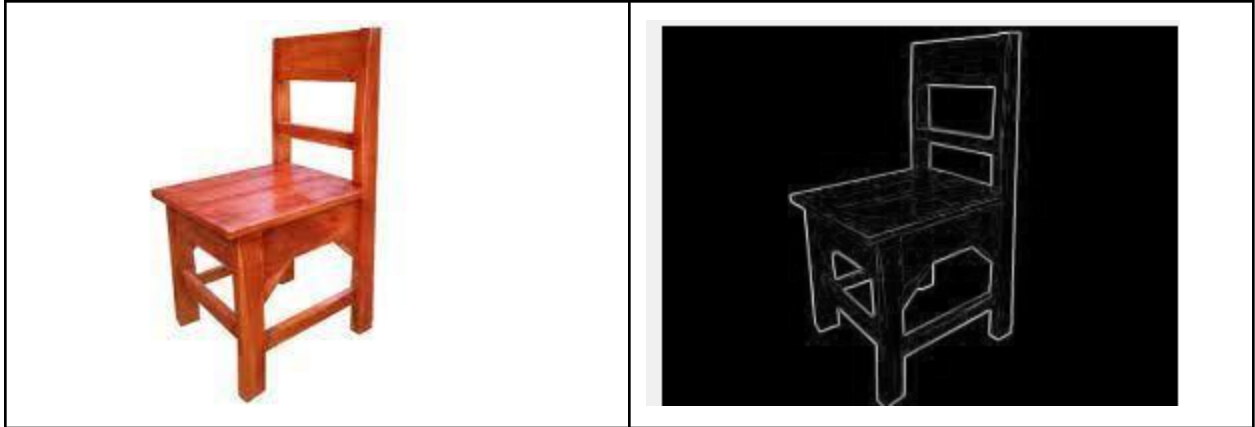
Kode Program

```
function result = robert(img)
    img = rgb2gray(img);
    Rx = [1 0;0 -1];
    Ry = [0 1;-1 0];
    Jx = convn(double(img), double(Rx), 'same');
    Jy = convn(double(img), double(Ry), 'same');
    result = uint8(sqrt(Jx.^2 + Jy.^2));
    %figure, imshow(result);
end
```

Hasil Eksekusi

Gambar	Hasil Deteksi Tepi
	
	
	





Analisis



Pada dasarnya mirip dengan algoritma Sobel, namun pada Robert, mask Rx dan Ry yang digunakan memiliki nilai yang sudah tetap.

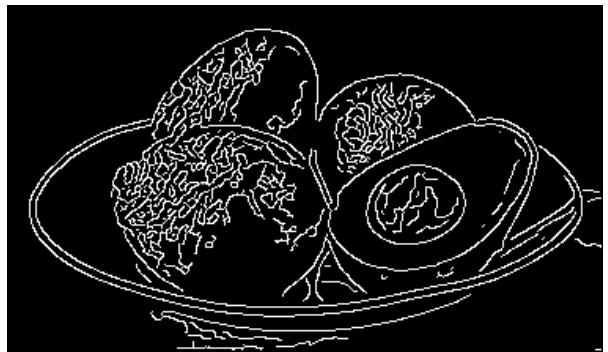
Task 6. Canny

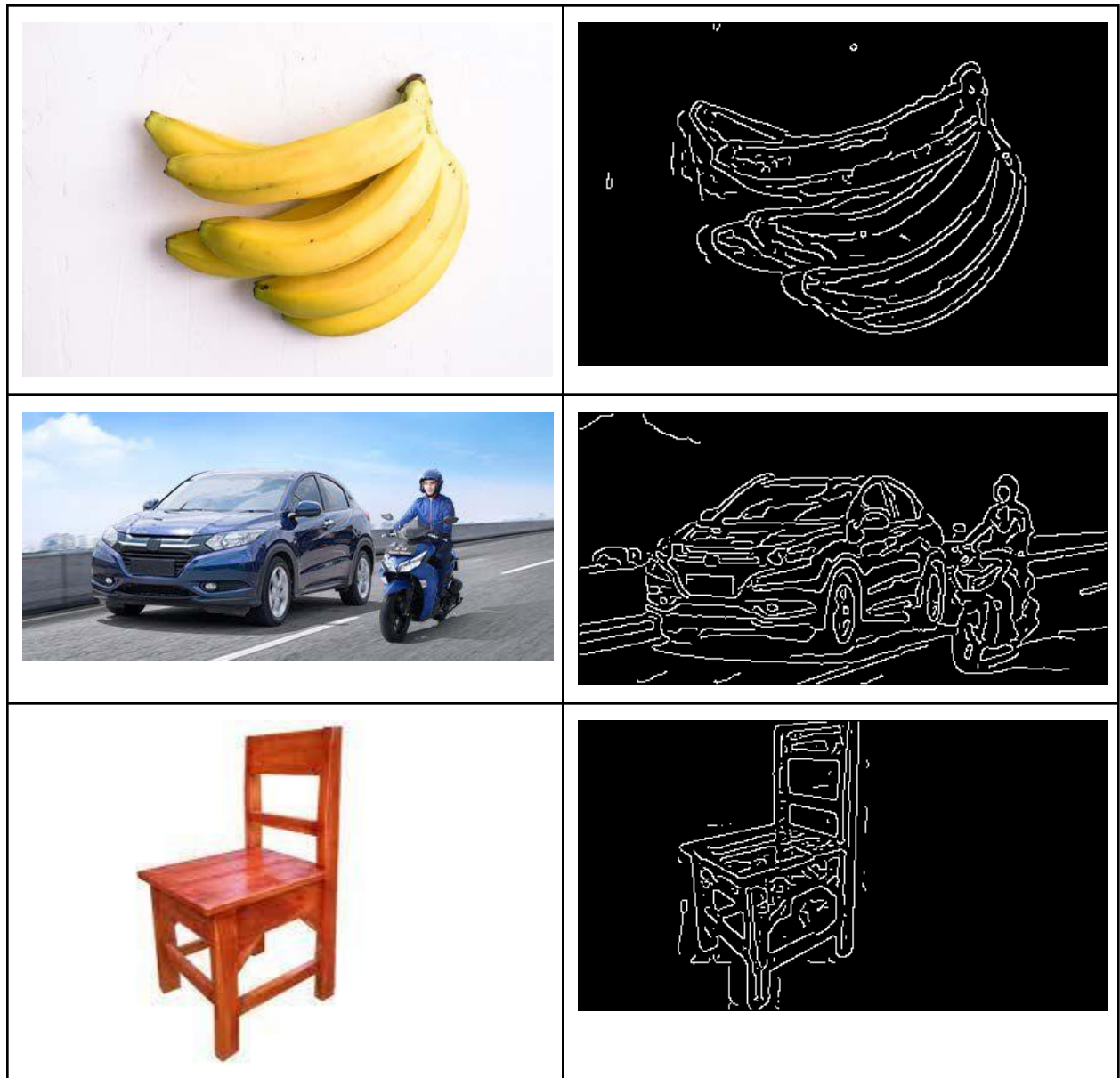
Kode Program

```
function result = canny(img)
    img = rgb2gray(img);
    result = edge(img, "canny");
end
```

Hasil Eksekusi

Gambar	Hasil Deteksi Tepi
	





Analisis

Operator canny merupakan operator edge detection yang paling umum digunakan. Operator canny menghasilkan edge dengan ketebalan 1 pixel dan citra yang dihasilkan merupakan citra biner. Operator ini ideal digunakan untuk object segmentation.

Task 7. Object Segmentation

Kode Program

```
function result = objectSegment(originalImg, edgedImg, k)

    [height, width, channels] = size(originalImg);
    closed_img = morphologicalClose(edgedImg, k);
    fill_img = imfill(closed_img, "holes");

    r = originalImg(:,:,1) .* uint8(fill_img);
    g = originalImg(:,:,2) .* uint8(fill_img);
    b = originalImg(:,:,3) .* uint8(fill_img);
    result = cat(3, r,g,b);
End

function result = morphologicalClose(img, k)
    result = dilation(img, k);
    result = erosion(result, k);
End

function result = dilation(img, k)
    % source : geeksforgeeks
    [height, width, ~] = size(img);
    result = zeros(height,width);

    % create structuring element
    se=ones(k, k);

    % store number of rows in P and
    % number of columns in Q.
    [P, Q]=size(se);

    % create a zero matrix of size I.
    result=zeros(height, width);

    for i=ceil(P/2):height-floor(P/2)
        for j=ceil(Q/2):width-floor(Q/2)

            % take all the neighbourhoods.
            on=img(i-floor(P/2):i+floor(P/2),
j-floor(Q/2):j+floor(Q/2));

            % take logical se
            nh=on(logical(se));

            % compare and take minimum value of the neighbor
```

```

        % and set the pixel value to that minimum value.
        result(i, j)=max(nh(:));
    end
end
end

function result = erosion(img, k)
    % source : geeksforgeeks
    [height, width, ~] = size(img);

    % create structuring element
    se=ones(k, k);

    % store number of rows in P and
    % number of columns in Q.
    [P, Q]=size(se);

    % create a zero matrix of size I.
    result=zeros(height, width);

    for i=ceil(P/2):height-floor(P/2)
        for j=ceil(Q/2):width-floor(Q/2)

            % take all the neighbourhoods.
            on=img(i-floor(P/2):i+floor(P/2),
j-floor(Q/2):j+floor(Q/2));

            % take logical se
            nh=on(logical(se));

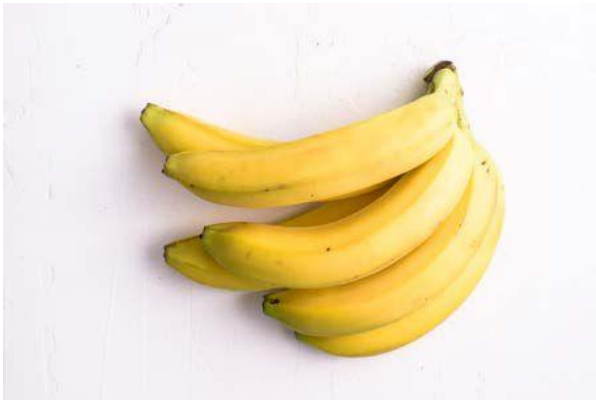
            % compare and take minimum value of the neighbor
            % and set the pixel value to that minimum value.
            result(i, j)=min(nh(:));
        end
    end
end
end

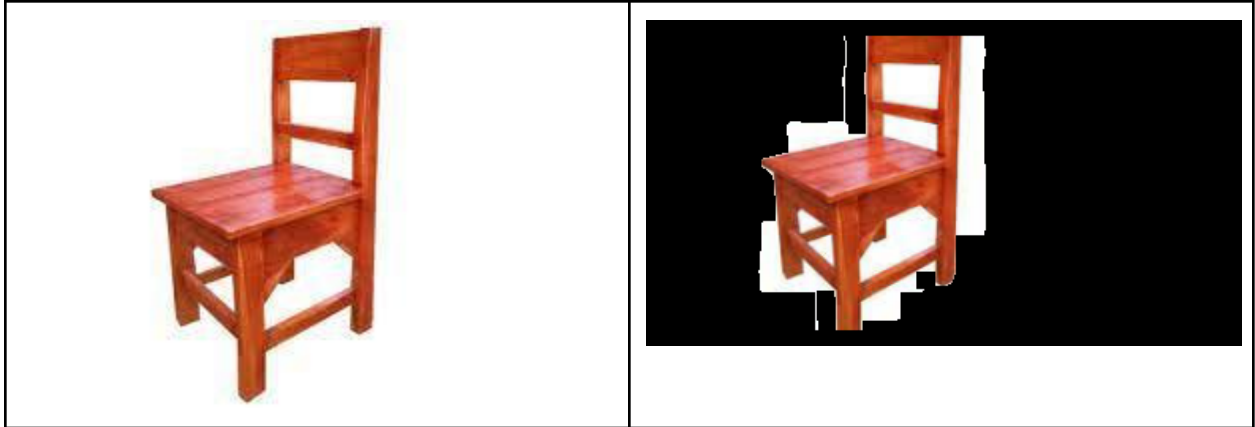
```

Hasil Eksekusi

Gambar	Hasil Segmentasi Objek
--------	------------------------







Analisis

Kami menggunakan Canny operator untuk deteksi tepi karena operator tersebut menghasilkan tepi yang jelas untuk dilakukan proses segmentasi objek. Berikut adalah urutan kami dalam melakukan segmentasi objek :

1. Deteksi tepi dengan operator canny
 Sebelum dilakukan deteksi tepi dengan operator canny, dilakukan blurring dengan gaussian filter. Hal ini digunakan untuk meminimalisir detail atau tepi yang tidak diinginkan, sehingga hasil dari operator canny diharapkan akan mendeteksi tepi pada objek tanpa detail yang banyak.
2. Melakukan *morphological close* pada gambar tepi.
 Morphology pada pemrosesan gambar adalah operasi pada gambar yang memproses gambar berdasarkan bentuknya. Terdapat dua operasi pada morphology, yaitu **dilation** dan **erosion**. Kode dilation dan erosion tidak buat kami sendiri karena berada di luar scope perkuliahan. Kami mendapat kode dilation [disini](#) dan erosion [disini](#). Dilation adalah proses untuk memperbesar pixel pada gambar biner sedangkan erosion adalah sebaliknya.
 Morphological close adalah operasi dilation yang diikuti dengan erosion pada gambar yang bertujuan untuk menyatukan edge pada gambar yang terpisah.
3. Melakukan imfill pada gambar
 Setelah tepi tersambung, maka akan dilakukan imfill, yaitu fungsi built-in pada matlab yang mengisi ruang yang terhubung garis dengan warna putih. Citra biner yang telah dilakukan imfill ini akan menjadi mask pada gambar original.
4. Melakukan perkalian gambar original dengan mask yang dihasilkan.

Metode ini tidak menghasilkan segmentasi objek yang tidak sempurna, namun berfungsi untuk mensegmentasi objek dari sebagian besar background. Metode ini berfungsi paling baik pada background gambar yang polos.

Alamat Github : <https://github.com/Haruray/tugas3-citra>