Tugas 3 IF4073 Interpretasi dan Pengolahan Citra

Semester I Tahun 2023/2024

Disusun oleh:

Fadil Fauzani	13520032
Mahesa Lizardy	13520116

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

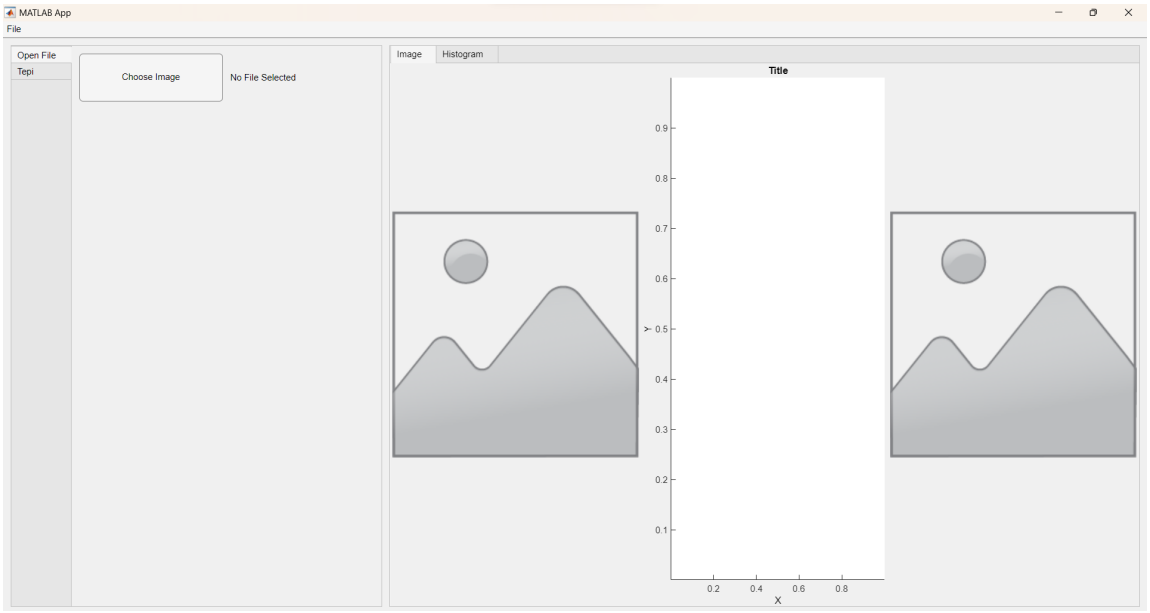
Jl. Ganesha 10, Bandung 40132

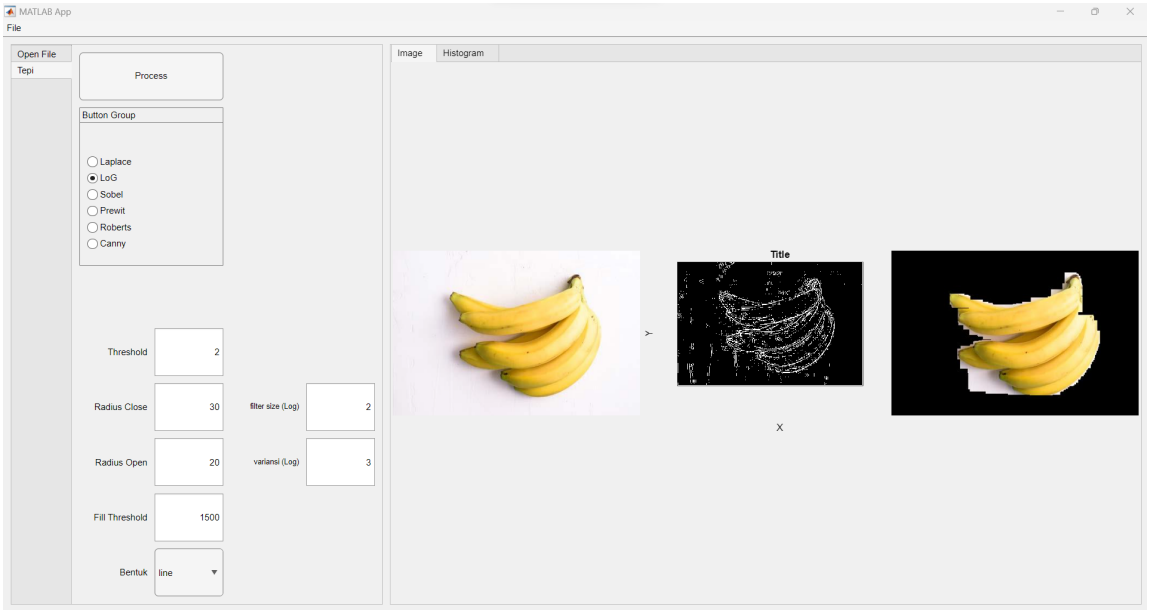
2023

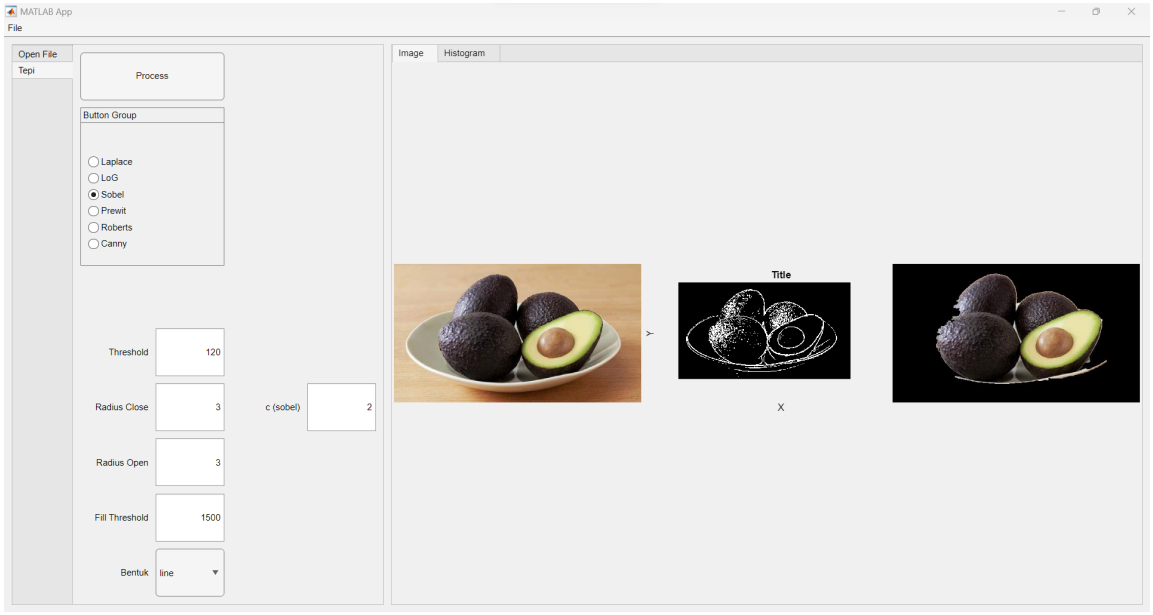
Daftar Isi

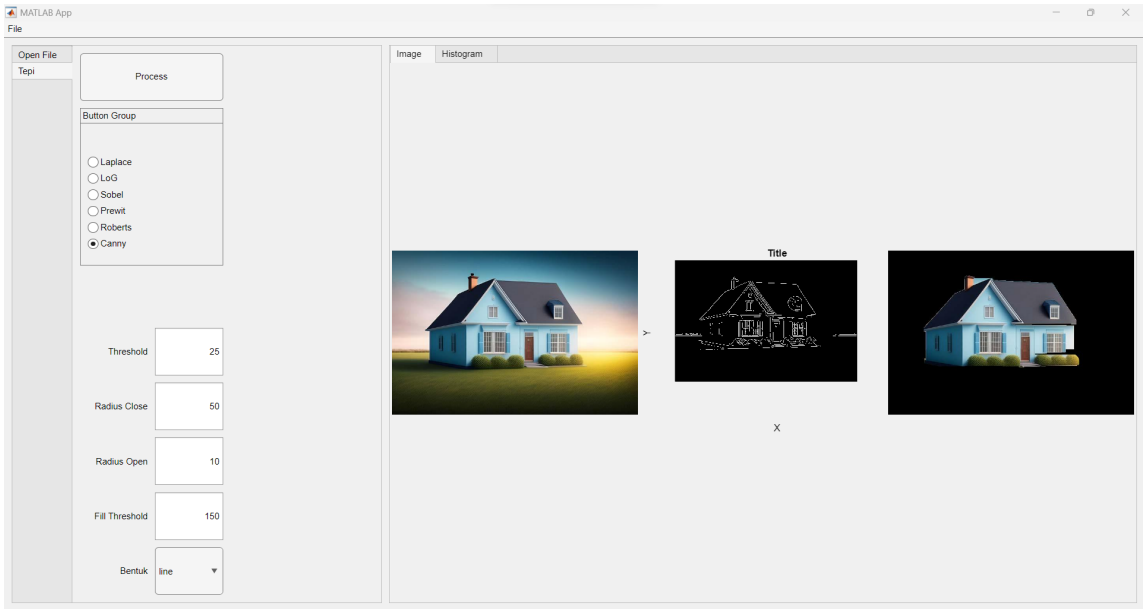
I GUI Program

Berikut adalah beberapa tangkapan layar GUI program. Sebelum memproses citra, pengguna perlu memasukkan gambar pada menu Open File atau dropdown toolbar “File > Open”. Pada tiap menu transformasi, terdapat tombol “Process”. Tombol “Process” digunakan untuk melakukan deteksi tepi objek menggunakan beberapa operator, setelah itu objek dipisahkan dari latar belakangnya yang hasilnya dapat dilihat pada tab “Image”.






MATLAB App
File
Open File
Tepi
Process
Button Group
Laplace
LoG
Sobel
Prewit
Roberts
Canny
Threshold
120
Radius Close
3
c (sobel)
2
Radius Open
3
Fill Threshold
1500
Bentuk
line
Image
Histogram
Title
X


MATLAB App
File
Open File
Tepi
Process
Button Group
Laplace
LoG
Sobel
Prewit
Roberts
Canny
Threshold
25
Radius Close
50
Radius Open
10
Fill Threshold
150
Bentuk
line
Image
Histogram
Title
X

II Pembahasan

2.1. Edge Detection

Pendeteksian tepi bertujuan untuk meningkatkan penampakan garis batas atau objek di dalam citra. Pendeteksian tepi mengekstraksi representasi gambar garis-garis di dalam citra. Pendeteksian tepi berguna dalam mengenali objek di dalam citra. Terdapat beberapa operasi yang dapat digunakan untuk melakukan pendeteksian tepi diantaranya Laplace, LoG, Sobel, Prewitt, Roberts, dan Canny.

2.1.1. Laplace

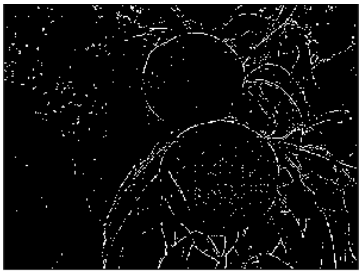
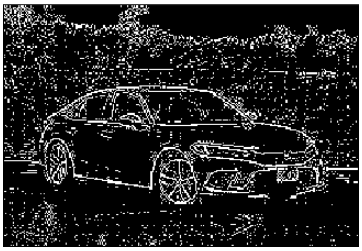
Operator Laplace dalam domain edge detection adalah alat yang digunakan dalam pengolahan citra untuk menemukan perubahan tajam dalam kecerahan antara piksel-piksel citra. Ini mencari tepi objek dengan konvolusi citra menggunakan kernel Laplace, sehingga menciptakan citra yang menyoroti tepi. Ambang batas kemudian dapat diterapkan untuk menghasilkan citra biner yang menampilkan tepi objek.

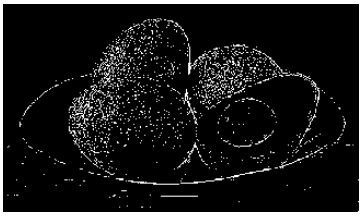
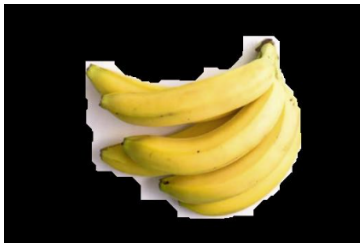
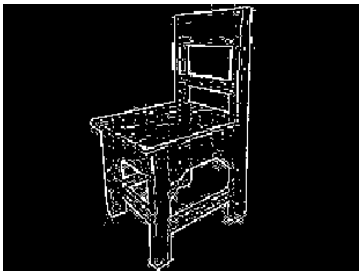
2.1.1.1. Kode Program

```
function result = laplace(grayImage, threshold)
 logMask = [0, -1, 0; -1, 4, -1; 0, -1, 0];
 result = conv2(double(grayImage), double(logMask), 'same');
 % thresholding to binary image
 result = uint8(result);
 result = result > threshold;
 % Remove pixel tapi to 0
 result([1 end], :) = 0; % Top and bottom rows
 result(:, [1 end]) = 0; % Left and right columns
end
```

2.1.1.2. Testing

Berikut testing untuk pendeteksian tepi dan segmentasi objek menggunakan operator Laplace

Input	Edge Detection	Image Segmentation
	 Threshold = 26	 Radius Close = 13 Radius Open = 40 Fill threshold = 1 Bentuk = Diamond
	 Threshold = 35	 Radius Close = 2 Radius Open = 25 Fill threshold = 1000 Bentuk = Disk
	 Threshold = 20	 Radius Close = 30 Radius Open = 10 Fill threshold = 1 Bentuk = Square
	 Threshold = 15	 Radius Close = 6 Radius Open = 2 Fill threshold = 1 Bentuk = Octagon

	 Threshold = 20	 Radius Close = 6 Radius Open = 15 Fill threshold = 1 Bentuk = Sphere
	 Threshold = 6	 Radius Close = 18 Radius Open = 20 Fill threshold = 1 Bentuk = Octagon
	 Threshold = 20	 Radius Close = 15 Radius Open = 20 Fill threshold = 1 Bentuk = Diamond
	 Threshold = 20	 Radius Close = 1 Radius Open = 5 Fill threshold = 1 Bentuk = Diamond

2.1.1.3. Analisis

Operator Laplace menghasilkan hasil pendeteksian yang kurang baik. Terlihat pada testing, Tepi yang dihasilkan masih banyak noise, bahkan tepi yang dihasilkan susah dipisahkan dengan noise yang ada walaupun dengan mengubah-ubah nilai threshold.

2.1.2. LoG

Metode Laplace of Gaussian (LoG) adalah teknik umum dalam pengolahan citra untuk mendeteksi tepi. Operasi Laplace dapat memunculkan tepi-tepi palsu karena noise dan variasi dalam citra. Salah satu cara untuk mengatasi masalah ini adalah menggabungkan operasi laplace dengan menapis citra terlebih dahulu dengan fungsi Gaussian.

2.1.2.1. Kode Program

Berikut kode program untuk operasi LoG.

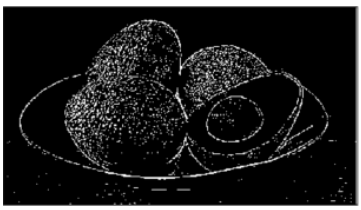
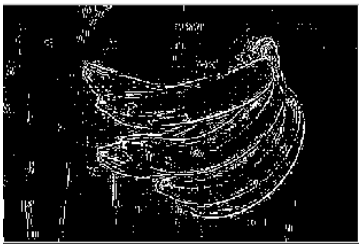
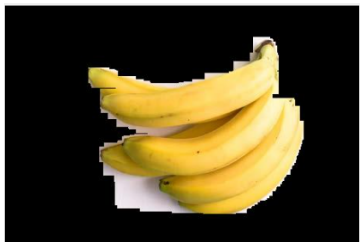
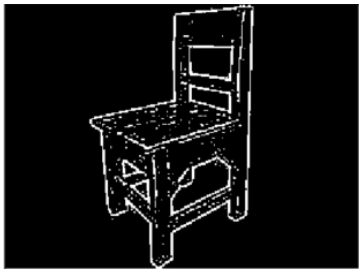
```
% Pendeteksian tepi dengan dengan operator laplace of gaussian
function result = laplaceOfGaussian(grayImage, size, sigma, threshold)
  gauss = fspecial('gaussian',size,sigma);
  result = conv2(double(grayImage), double(gauss), 'same');
  % aktifkan dengan filter gauss lalu masukkan ke laplace
  laplace = [0, -1, 0; -1, 4, -1; 0, -1, 0];
  result = conv2(double(result), double(laplace), 'same');
  % thresholding to binary image
  result = uint8(result);
  result = result > threshold;
  % Change pixel tepi to 0 untuk menghindari kesalahan image fill
  result([1 end], :) = 0; % Top and bottom rows
  result(:, [1 end]) = 0; % Left and right columns
end
```

2.1.2.2. Testing

Berikut testing untuk pendeteksian tepi dan segmentasi objek menggunakan operator LoG

Input	Edge Detection	Image Segmentation

	 Threshold = 6 Filter size (log) = 11 Variansi (log) = 1	 Radius Close = 4 Radius Open = 90 Fill Threshold =5000 Bentuk = sphere
	 Threshold = 2 Filter size (log) = 6 Variansi (log) = 1	 Radius Close = 1 Radius Open = 1 Fill Threshold =7000 Bentuk = line
	 Threshold = 5 Filter size (log) = 6 Variansi (log) = 1	 Radius Close = 35 Radius Open = 100 Fill Threshold =10 Bentuk = square
	 Threshold = 5 Filter size (log) = 2 Variansi (log) = 1	 Radius Close = 4 Radius Open = 100 Fill Threshold =10 Bentuk = line

	 Threshold = 14 Filter size (log) = 2 Variansi (log) = 1	 Radius Close = 4 Radius Open = 45 Fill Threshold =1 Bentuk = sphere
	 Threshold = 2 Filter size = 2 Variansi (LoG) = 2	 Radius Close = 20 Radius Open = 30 Fill Threshold = 1500 Bentuk = line
	 Threshold = 1 Filter size = 2 Variansi (LoG) = 1	 Radius Close = 16 Radius Open = 100 Fill Threshold = 1 Bentuk = line
	 Threshold = 13 Filter size = 2 Variansi (LoG) = 1	 Radius Close = 3 Radius Open = 10 Fill Threshold = 1 Bentuk = line

2.1.2.3. Analisis

Operator LoG cukup baik untuk mendeteksi tepi objek dan memisahkan dari latar belakangnya. Namun, terdapat beberapa citra yang tidak dapat dihasilkan secara baik seperti pada citra mobil & motor yang mengandung noise tinggi sehingga menghasilkan tepi yang kabur dan tidak akurat. Hal ini disebabkan LoG sangat sensitif terhadap noise dalam citra.

2.1.3. Sobel

Operator Sobel adalah sebuah teknik dalam pengolahan citra digital yang digunakan untuk mendeteksi tepi pada gambar. Operasi ini bekerja dengan menghitung gradien citra menggunakan kernel Sobel, yang terdiri dari dua matriks konvolusi, satu untuk mendeteksi perubahan intensitas horizontal dan yang lainnya untuk mendeteksi perubahan intensitas vertikal.

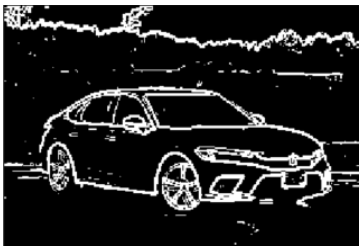
2.1.3.1. Kode Program

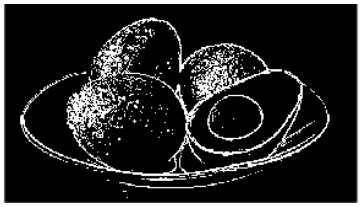
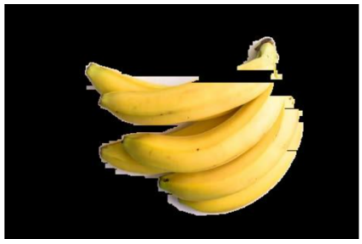
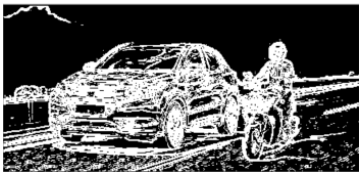
Berikut kode program untuk operasi sobel.

```
% Pendeteksian tepi dengan dengan operator sobel
function result = sobel(grayImage, c, threshold)
  % Sobel Kernels
  sobelHorizontal = [-1, 0, 1; -c, 0, c; -1, 0, 1];
  sobelVertical = [1, c, 1; 0, 0, 0; -1, -c, -1];
  % Convolve image with Sobel kernels
  gradientX = conv2(double(grayImage), double(sobelHorizontal), 'same');
  gradientY = conv2(double(grayImage), double(sobelVertical), 'same');
  % Gradient Magnitude and Direction
  result = uint8(sqrt(gradientX.^2 + gradientY.^2));
  result = result > threshold;
  % Change pixel tepi to 0 untuk menghindari kesalahan image fill
  result([1 end], :) = 0; % Top and bottom rows
  result(:, [1 end]) = 0; % Left and right columns
end
```

2.1.3.2. Testing

Berikut testing untuk pendeteksian tepi dan segmentasi objek menggunakan operator Sobel

Input	Edge Detection	Image Segmentation
	 Threshold = 210 C (sobel) = 2	 Radius Close = 100 Radius Open = 100 Fill Threshold = 5000 Bentuk = line
	 Threshold = 200 C (sobel) = 2	 Radius Close = 90 Radius Open = 40 Fill Threshold = 5000 Bentuk = line
	 Threshold = 150 C (sobel) = 4	 Radius Close = 100 Radius Open = 30 Fill Threshold = 1 Bentuk = line

	 Threshold = 55 C (sobel) = 3	 Radius Close = 1 Radius Open = 1 Fill Threshold = 1 Bentuk = line
	 Threshold = 114 C (sobel) = 2	 Radius Close = 114 Radius Open = 3 Fill Threshold = 10 Bentuk = line
	 Threshold = 80 C (sobel) = 4	 Radius Close = 100 Radius Open = 5 Fill Threshold = 10 Bentuk = line
	 Threshold = 70 C (sobel) = 3	 Radius Close = 2 Radius Open = 17 Fill Threshold = 1000 Bentuk = line

	 Threshold = 25 C (sobel) = 2	 Radius Close = 3 Radius Open = 15 Fill Threshold = 1 Bentuk = line

2.1.3.3. Analisis

Operator sobel cukup baik untuk mendeteksi tepi objek dan memisahkan dari latar belakangnya.

2.1.4. Prewitt

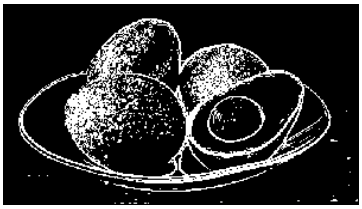
Operator Prewitt adalah bekerja dengan penapis yang sama dengan operator sobel tetapi dengan nilai konstanta C = 1.

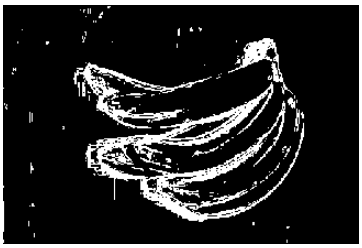
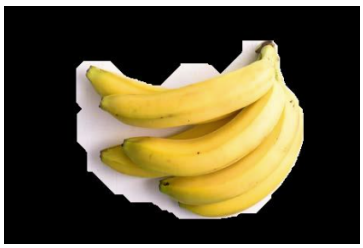
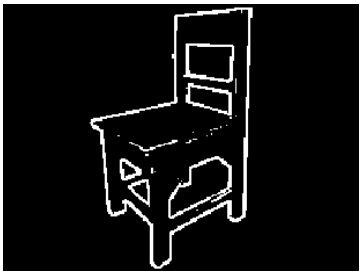
2.1.4.1. Kode Program

```
% Pendeteksian tepi dengan dengan operator priwit
function result = prewit(grayImage, threshold)
  result = sobel(grayImage, 1, threshold);
end
```

2.1.4.2. Testing

Input	Edge Detection	Image Segmentation
	 Threshold = 140	 Radius Close = 15 Radius Open = 111

		Fill Threshold = 1000 Bentuk = Diamond
	 Threshold = 70	 Radius Close = 3 Radius Open = 8 Fill Threshold = 1000 Bentuk = line
	 Threshold = 25	 Radius Close = 3 Radius Open = 15 Fill Threshold = 1000 Bentuk = line
	 Threshold = 40	b Radius Close = 1 Radius Open = 1 Fill Threshold = 1 Bentuk = line
	 Threshold = 65	 Radius Close = 1 Radius Open = 15 Fill Threshold = 1 Bentuk = Diamond

	 Threshold = 25	 Radius Close = 25 Radius Open = 35 Fill Threshold = 1 Bentuk = Diamond
	 Threshold = 50	 Radius Close = 3 Radius Open = 12 Fill Threshold = 1 Bentuk = line
	 Threshold = 160	 Radius Close = 1 Radius Open = 1 Fill Threshold = 1 Bentuk = line

2.1.4.3. Analisis

Operator Prewitt cukup baik untuk mendeteksi tepi objek. Namun, untuk latar belakangnya, operator ini susah menyaring noise yang ada, terlihat pada gambar mobil-motor, mobil, dan rumah dimana pada gambar tersebut terdeteksi banyak titik-titik noise pada hasil *edge-detection.*

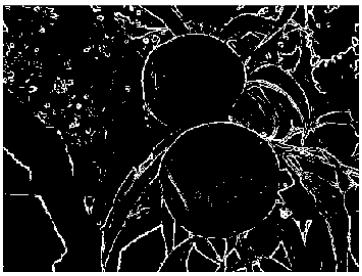
2.1.5. Roberts

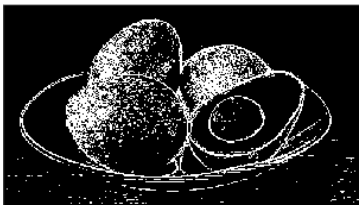
Operator Roberts adalah operator yang digunakan dalam pemrosesan citra untuk deteksi tepi. Operator ini terdiri dari dua kernel, yaitu kernel Roberts-1 dan kernel Roberts-2. Kedua kernel ini digunakan untuk menghitung gradien citra dalam arah diagonal. Hasil dari operasi ini dapat digunakan untuk mengidentifikasi perubahan tajam dalam intensitas citra, yang sering kali menandakan adanya tepi atau batas objek dalam citra. Operator Roberts bekerja dengan mengaplikasikan konvolusi pada citra asli dengan kedua kernel ini, dan hasilnya digunakan untuk mengidentifikasi tepi dalam citra dalam arah diagonal.

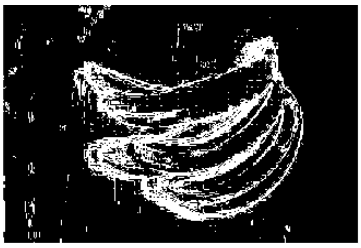
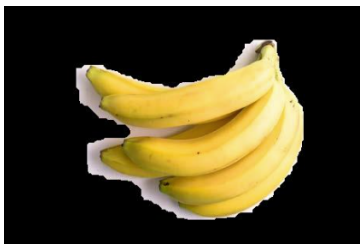
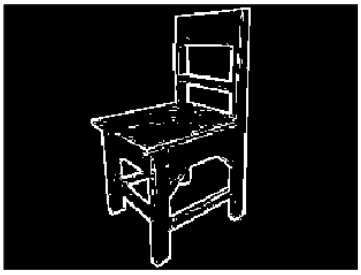
2.1.5.1. Kode Program

```
function result = roberts(grayImage,threshold)
  Rx = [1, 0; 0, -1];
  Ry = [0, 1; -1, 0];
  Jx = conv2(double(grayImage),double(Rx),'same');
  Jy = conv2(double(grayImage),double(Ry),'same');
  Jedge = sqrt(Jx.^2 + Jy.^2);
  result = uint8(Jedge);
  result = result > threshold;
end
```

2.1.5.2. Testing

Input	Edge Detection	Image Segmentation
	 Threshold = 40	 Radius Close = 14 Radius Open = 25 Fill Threshold = 1000 Bentuk = diamond

	 Threshold = 75	 Radius Close = 10 Radius Open = 10 Fill Threshold = 1000 Bentuk = disk
	 Threshold = 10	 Radius Close = 5 Radius Open = 60 Fill Threshold = 1 Bentuk = Square
	 Threshold = 30	 Radius Close = 1 Radius Open = 1 Fill Threshold = 1 Bentuk = Square
	 Threshold = 22	 Radius Close = 1 Radius Open = 1 Fill Threshold = 1 Bentuk = line

	 Threshold = 5	 Radius Close = 4 Radius Open = 25 Fill Threshold = 1000 Bentuk = Diamond
	 Threshold = 35	 Radius Close = 40 Radius Open = 15 Fill Threshold = 1 Bentuk = Square
	 Threshold = 35	 Radius Close = 1 Radius Open = 1 Fill Threshold = 1 Bentuk = Line

2.1.5.3. Analisis

Operator Roberts cukup baik untuk mendeteksi tepi objek dan tepi latar belakangnya. Namun, untuk gambar yang latar belakangnya "ramai", tepi yang ada di latar belakang juga terdeteksi, terlihat pada gambar mobil-motor, rumah, dan buah.

2.1.6. Canny

Operator Canny adalah metode deteksi tepi multistep yang melibatkan beberapa tahap pengolahan citra, termasuk penghalusan (smoothing) dengan filter Gaussian, perhitungan

gradien menggunakan operator Sobel, penemuan tepi yang akurat dengan menggunakan histeresis, dan pemilihan tepi utama.

2.1.6.1. Kode Program

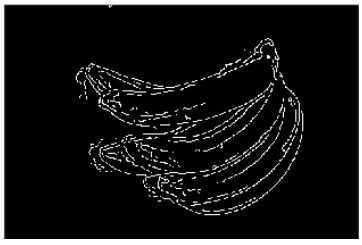
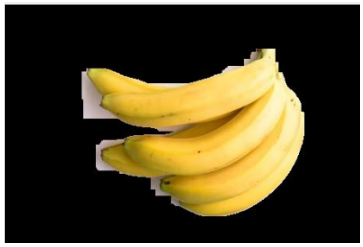
Berikut kode program untuk operasi canny menggunakan fungsi built in matlab edge.

```
% Pendeteksian tepi dengan dengan operator canny
function result = canny(I, threshold)
  result = edge(I,"canny", threshold);
end
```

2.1.6.2. Testing

Berikut testing untuk pendeteksian tepi dan segmentasi objek menggunakan operator Sobel

Input	Edge Detection	Image Segmentation
	 Threshold = 102	 Radius Close = 14 Radius Open = 15 Fill Threshold = 4000 Bentuk = line
	 Threshold = 65	 Radius Close = 70 Radius Open = 5 Fill Threshold = 4000 Bentuk = line

	 Threshold = 25	 Radius Close = 50 Radius Open = 10 Fill Threshold = 150 Bentuk = line
	 Threshold = 25	 Radius Close = 50 Radius Open = 10 Fill Threshold = 150 Bentuk = line
	 Threshold = 80	 Radius Close = 5 Radius Open = 10 Fill Threshold = 1 Bentuk = sphere
	 Threshold = 22	 Radius Close = 16 Radius Open = 50 Fill Threshold = 15 Bentuk = line

	 Threshold = 40	 Radius Close = 2 Radius Open = 10 Fill Threshold = 1000 Bentuk = sphere
	 Threshold = 100	 Radius Close = 3 Radius Open = 3 Fill Threshold = 1 Bentuk = sphere

2.1.6.3. Analisis

Operator canny cukup baik untuk mendeteksi tepi objek dan memisahkan dari latar belakangnya.

2.2. Pemisahan Latar Belakang

Pemisahan latar belakang dilakukan untuk memisahkan objek dari latar belakang. Operasi ini dilakukan setelah operasi *edge detection*.

2.2.1. Kode Program

Berikut kode program untuk operasi pemisahan latar belakang objek.

```
radiusClose = app.RadiusCloseEditField.Value;
radiusOpen = app.RadiusOpenEditField.Value;
fill_tresh = app.FillThresholdEditField.Value;
imshow(app.Outputimagedata, 'Parent',app.EdgeImage);
% operasi penutupan citra
```

```
if (strcmp(app.BentukDropDown.Value,'line'))
  result = imclose(app.Outputimagedata , strel(app.BentukDropDown.Value, radiusClose, 0));
else
  result = imclose(app.Outputimagedata , strel(app.BentukDropDown.Value, radiusClose));
end
  % operasi filling holes
  result = imfill(result,'holes');
  % menghilangkan noise dan detail detail kecil
  result = imopen(result,strel(ones(radiusOpen,radiusOpen)));
  % menghapus objek-objek kecil objek-objek luar kurang dari
  result = bwareaopen(result, fill_tresh);
  % operasi perkalian untuk masing masing kernel untuk
if (numChannels == 1)
  gray_processed=app.Inputimagedata(:,:,1).*uint8(result);
  app.Outputimagedata=cat(3,gray_processed,gray_processed,gray_processed);
elseif (numChannels == 3)
  red_processed=app.Inputimagedata(:,:,1).*uint8(result);
  green_processed=app.Inputimagedata(:,:,2).*uint8(result);
  blue_processed=app.Inputimagedata(:,:,3).*uint8(result);
  app.Outputimagedata=cat(3,red_processed,green_processed,blue_processed);
else
% Handle other cases (e.g., images with more than 3 channels)
  msgbox('Unsupported image format', 'Error', 'error');
end
% show image
numChannels = size(app.Outputimagedata, 3);
app.setHistogramOutput(numChannels);
```

2.2.2. Analisis

Pemisahan latar belakang memanfaatkan beberapa variabel, seperti Radius Close, Radius Open, Fill Threshold, dan Bentuk, yang dapat disesuaikan oleh pengguna. Pengaturan variabel-variabel ini secara langsung mempengaruhi kualitas hasil pemisahan latar belakang.

Alamat GitHub Program

https://github.com/lizardyy/Tugas-3-Citra

Referensi

PPT IF4073 Interpretasi dan Pengolahan Citra 2023/2024 (https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2023-2024/citra23-24.htm).