**TUGAS 3**

**IF4073 Pemrosesan Citra Digital**

**A person in a black jacket

Description automatically generated**

Oleh:

13520052 – Gregorius Moses Marevson

1. Screenshot GUI

Gambar di bawah adalah tampilan awal dari GUI yang telah dibuat.

A screenshot of a computer

Description automatically generated

1. Untuk setiap program ditampilkan kode, hasil eksekusi, dan analisis sebagai berikut:
   1. Kode program
2. Edge Detection

|  |
| --- |
| detectEdge.m  function edgeImage = detectEdge(image, type)  image = im2gray(image);  if type == "canny"  edgeImage = edge(image, 'canny');  return  end  image = double(image);  switch type  case 'laplace'  mask = [0 1 0; 1 -4 1; 0 1 0];  edgeImage = conv2(image, mask, 'same');  case 'log'  mask = fspecial('log');  edgeImage = conv2(image, mask, 'same');  case {'sobel', 'prewitt', 'roberts'}  edgeImage = doGradientBasedEdgeDetection(image, type);  end  edgeImage = uint8(edgeImage);  edgeImage = im2double(edgeImage) > graythresh(edgeImage);  end  function edgeImage = doGradientBasedEdgeDetection(image, type)  switch type  case 'sobel'  Gx = [-1 0 1; -2 0 2; -1 0 1];  Gy = [1 2 1; 0 0 0; -1 -2 -1];  case 'prewitt'  Gx = [-1 0 1; -1 0 1; -1 0 1];  Gy = [1 1 1; 0 0 0; -1 -1 -1];  case 'roberts'  Gx = [1 0; 0 -1];  Gy = [0 1; -1 0];  end  Jx = conv2(image, Gx, 'same');  Jy = conv2(image, Gy, 'same');  edgeImage = sqrt(Jx.^2 + Jy.^2);  end |

1. Object Detection

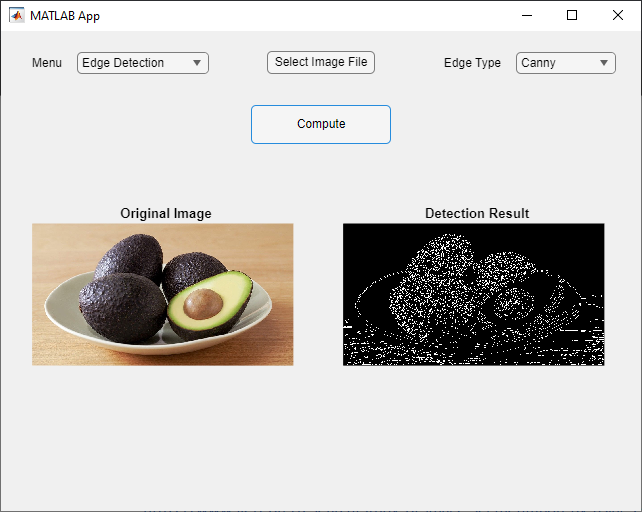
|  |
| --- |
| detectObject.m  function detectedObjects = detectObject(edgeImage, image)  % Measure properties for each object (connected component) in an image  % properties: "Area", "Centroid", and "BoundingBox"  % https://www.mathworks.com/help/images/ref/regionprops.html  stats = regionprops(edgeImage);    % Filter out small noise  validObjects = stats([stats.Area] > 100);  detectedObjects = zeros(size(edgeImage));  for i = 2:length(validObjects)  % BoundingBox: Position and size of the smallest box containing the region  % bbox: (x, y, width, height)  bbox = round(validObjects(i).BoundingBox);  % Create box for object  detectedObjects(bbox(2):bbox(2)+bbox(4), bbox(1):bbox(1)+bbox(3)) = 1;  end  % Display the object in original image  detectedObjects = uint8(detectedObjects .\* double(image));  end |

1. Line Detection

|  |
| --- |
| detectLines.m  function detectLines(image, useInBuilt, type, axesHandle)  % https://www.mathworks.com/help/images/ref/houghlines.html  % Create Hough transform using edge image  edgeImage = detectEdge(image, type);  if useInBuilt  houghFn = @hough;  else  houghFn = @houghTransform;  end  [H, T, R] = houghFn(edgeImage);  % Find peaks in the Hough transform  P = houghpeaks(H, 5, 'Threshold', ceil(0.3\*max(H(:))));  % Fine lines and plot them  lines = houghlines(edgeImage, T, R, P);  if nargin < 4  imshow(image), hold on  else  imshow(image, 'Parent', axesHandle), hold(axesHandle, "on")  end  for k = 1:length(lines)  % Extract line endpoints  xy = [lines(k).point1; lines(k).point2];    % Plot line segments  if nargin < 4  plot(xy(:,1), xy(:,2), 'LineWidth', 2, 'Color', 'r');  else  plot(axesHandle, xy(:,1), xy(:,2), 'LineWidth', 2, 'Color', 'r');  end    end  if nargin < 4  hold off  else  hold(axesHandle, "off")  end  end |
| houghTransform.m  function [parameter, theta, rho] = houghTransform(edgeImage, p, q)  % input : edgeImage with size m x n  % output: parameter with size p x q, theta, rho  % p : size of theta  % q : size of r  [m, n] = size(edgeImage);  sqrtd = sqrt(m^2 + n^2);  if nargin < 3  p = min(180, 2 \* ceil(max(m, n) \* sqrt(2)));  q = 2 \* floor(sqrtd) - 1;  end  parameter = zeros(p, q);  % -phi/2 <= theta <= phi/2  theta = linspace(-90, 90, p);  sin = sind(theta);  cos = cosd(theta);  rho = linspace(-sqrtd, sqrtd, q);  [x, y] = find(edgeImage);  for t=1:length(x)  i = x(t) - 1;  j = y(t) - 1;  for k=1:p  r = i\*sin(k) + j\*cos(k);  [~, ri] = min(abs(rho - r));  parameter(k, ri) = parameter(k, ri) + 1;  end  end  parameter = parameter';  end |
| inverseHough.m  function image = inverseHough(edge, parameter)  % input : edge with size m x n  % parameter with size p x q  % output: image with straight line  [m, n] = size(edge);  [p, q] = size(parameter);  image = zeros(m, n);  sin = sind(linspace(-90, 90, p));  cos = cosd(linspace(-90, 90, p));  sqrtd = sqrt(m^2 + n^2);    for i=1:p  for j=1:q  y = 0;  if parameter(i, j)  for k=1:m  r = j\*(2\*sqrtd)/(m-1) - sqrtd;  if sin(i) == 0  y = y + 1;  else  y = (r - k \* cos(i))/sin(i);  end  y = y + 1;  l = floor(y);  if (l >= 1 && l <= n)  if edge(k, l)  image(k, l) = image(k, l) + 1;  end  end  end  end  end  end  end |

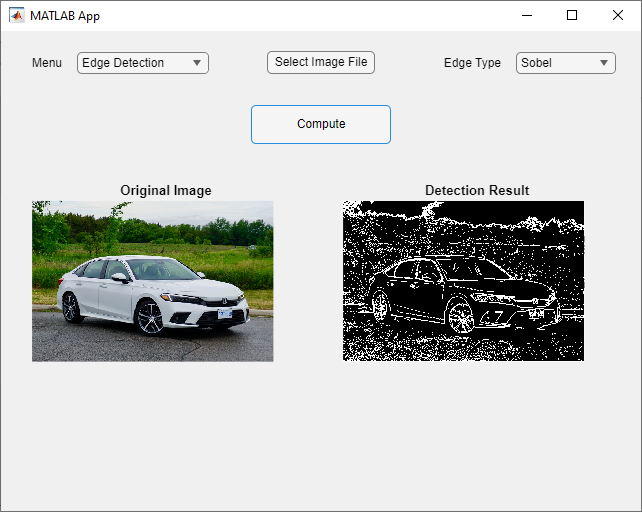
* 1. Hasil eksekusi

1. Edge Detection



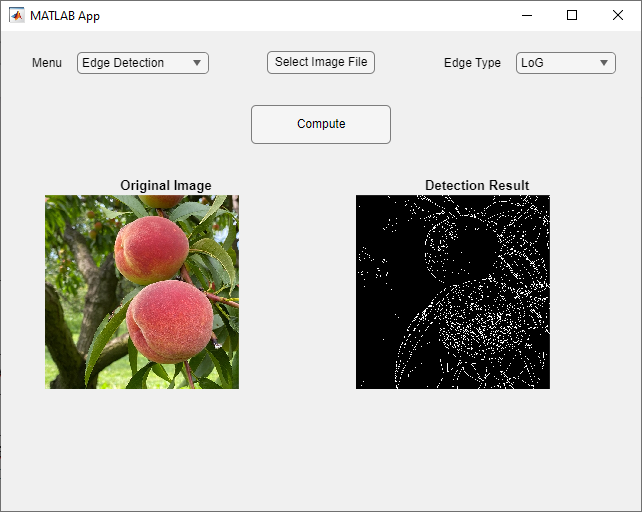
A screenshot of a computer

Description automatically generated



A screenshot of a computer

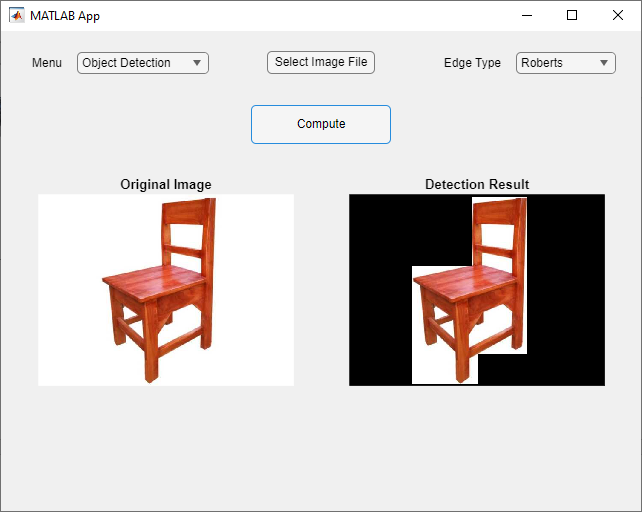
Description automatically generated



A screenshot of a computer

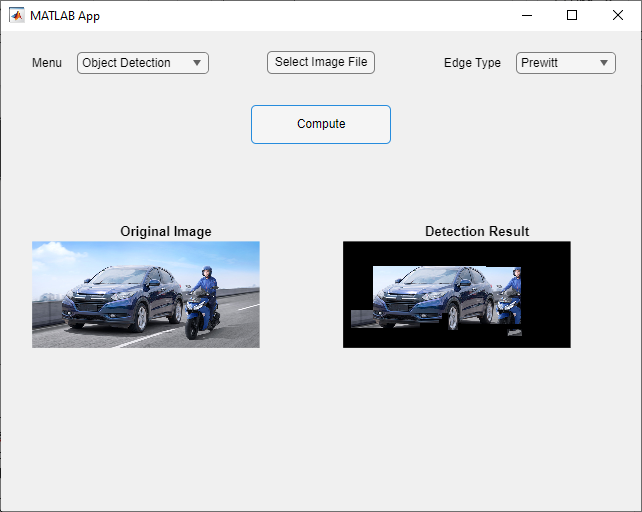
Description automatically generated

1. Object Detection



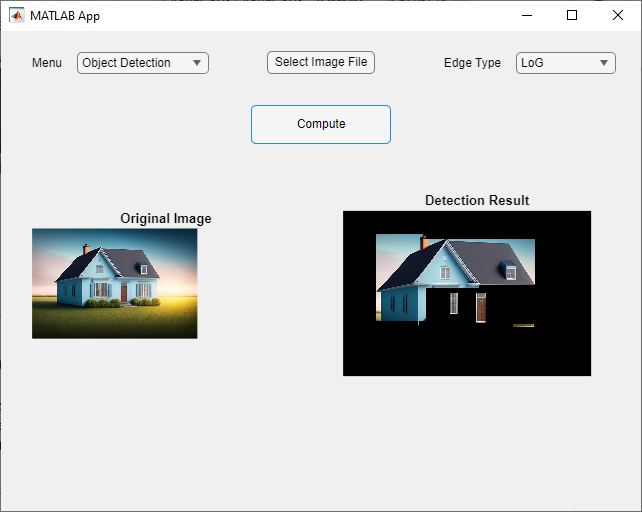
A screenshot of a computer

Description automatically generated



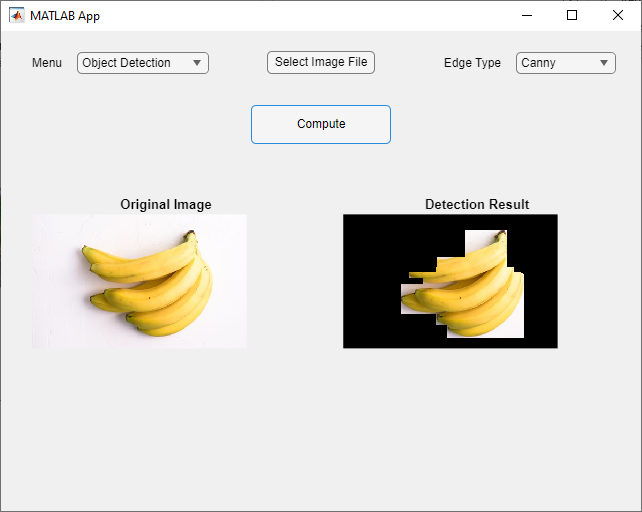
A screenshot of a computer

Description automatically generated



A screenshot of a computer

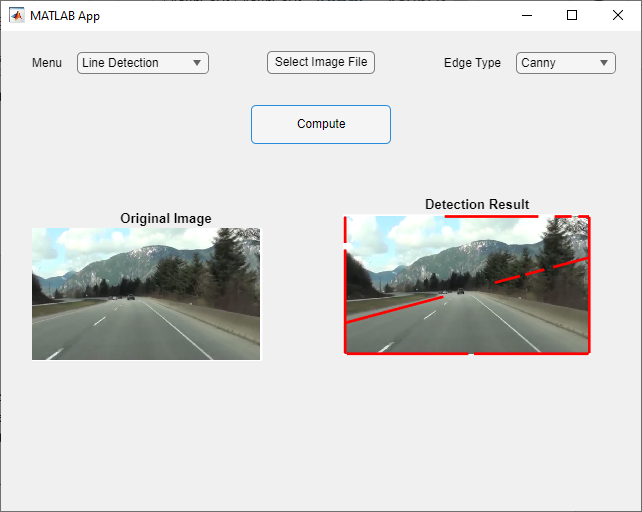
Description automatically generated



A screenshot of a computer

Description automatically generated

1. Line Detection



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* 1. Analisis

1. Edge Detection

Cara kerja program adalah: melakukan konvolusi antara mask dengan gambar pada algoritma laplace dan LoG, atau melakukan konvolusi antara mask pada sumbu x dan y, kemudian menghitung magnitude di gradien yang telah dihitung pada algoritma sobel, roberts, prewitt, dan canny. Implementasi canny menggunakan fungsi edge yang tersedia di matlab. Di akhir program, dilakukan global thresholding dengan algoritma otsu.

Dilihat dari hasil program, algoritma edge detection tidak memiliki masalah (bug).

1. Object Detection

Deteksi objek dengan menerima parameter edge dan gambar original. Pencarian objek dilakukan dengan fungsi regionprops yang tersedia di matlab. Regionprops menghasilkan sejumlah objek yang berhasil dideteksi dari edge, juga memberikan beberapa property seperti “Area”, “Centroid”, dan “BoundingBox”. Kemudian, dilakukan filtering untuk menghilangkan noise yaitu objek dengan area yang kecil. Untuk setiap objek yang dideteksi, dilakukan penghitungan kotak yang akan menunjukkan lokasi objek. Kotak dihasilkan dari BoundingBox (x, y, width, height), dengan mengisi gambar dengan nilai 1 pada range (x:x+width, y+y+height). Box tersebut kemudian dikalikan dengan gambar asli untuk menampilkan objek yang berhasil dideteksi.

Dilihat dari hasil program, algoritma object detection masih bisa diperbaiki, terutama dengan memanfaatkan fungsi imfill untuk mengisi daerah edge tertutup. Selain itu, deteksi objek masih bisa salah di beberapa gambar.

1. Line Detection

Deteksi garis pada gambar dilakukan menggunakan fungsi hough yang tersedia di matlab atau fungsi houghTransform yang dibuat sendiri. Untuk kecepatan algoritma, fungsi hough jauh lebih cepat dibanding houghTransform karena algoritma yang lebih effisien seperti penggunaan accumarray. Hough dan houghTransform menghasilkan parameter hough, theta, dan rho. Parameter hough kemudian digunakan untuk menghitung peaks dengan fungsi houghPeaks. Kemudian dilakukan pencarian garis menggunakan fungsi houghlines. Pencarian garis sudah diimplementasi sendiri pada fungsi inverseHough, tetapi belum menghasilkan hasil yang tepat dan membutuhkan waktu yang lama. Ditampilkan gambar original, kemudian dilakukan plot segmen garis untuk setiap garis pada lines yang dihasilkan houghlines.

Dilihat dari hasil program, algoritma line detection masih bisa dikembangkan lagi. Algoritma inverseHough yang telah dibuat juga dapat di refactor sehingga memiliki hasil yang dapat digunakan untuk melakukan plot.

1. Github

<https://github.com/Moses-ui/tugas-3-citra>