**01 Matlab Code**

1. **Hough.m**

function [h, theta, rho] = hough(f, dtheta, drho)

%HOUGH Hough transform.

% [H, THETA, RHO] = HOUGH(F, DTHETA, DRHO) computes the Hough

% transform of the image F. DTHETA specifies the spacing (in

% degrees) of the Hough transform bins along the theta axis. DRHO

% specifies the spacing of the Hough transform bins along the rho

% axis. H is the Hough transform matrix. It is NRHO-by-NTHETA,

% where NRHO = 2\*ceil(norm(size(F))/DRHO) - 1, and NTHETA =

% 2\*ceil(90/DTHETA). Note that if 90/DTHETA is not an integer, the

% actual angle spacing will be 90 / ceil(90/DTHETA).

%

% THETA is an NTHETA-element vector containing the angle (in

% degrees) corresponding to each column of H. RHO is an

% NRHO-element vector containing the value of rho corresponding to

% each row of H.

%

% [H, THETA, RHO] = HOUGH(F) computes the Hough transform using

% DTHETA = 1 and DRHO = 1.

% Copyright 2002-2004 R. C. Gonzalez, R. E. Woods, & S. L. Eddins

% Digital Image Processing Using MATLAB, Prentice-Hall, 2004

% $Revision: 1.4 $ $Date: 2003/10/26 22:33:44 $

if nargin < 3

drho = 1;

end

if nargin < 2

dtheta = 1;

end

f = double(f);

[M,N] = size(f);

theta = linspace(-90, 0, ceil(90/dtheta) + 1);

theta = [theta -fliplr(theta(2:end - 1))];

ntheta = length(theta);

D = sqrt((M - 1)^2 + (N - 1)^2);

q = ceil(D/drho);

nrho = 2\*q - 1;

rho = linspace(-q\*drho, q\*drho, nrho);

[x, y, val] = find(f);

x = x - 1; y = y - 1;

% Initialize output.

h = zeros(nrho, length(theta));

% To avoid excessive memory usage, process 1000 nonzero pixel

% values at a time.

for k = 1:ceil(length(val)/1000)

first = (k - 1)\*1000 + 1;

last = min(first+999, length(x));

x\_matrix = repmat(x(first:last), 1, ntheta);

y\_matrix = repmat(y(first:last), 1, ntheta);

val\_matrix = repmat(val(first:last), 1, ntheta);

theta\_matrix = repmat(theta, size(x\_matrix, 1), 1)\*pi/180;

rho\_matrix = x\_matrix.\*cos(theta\_matrix) + ...

y\_matrix.\*sin(theta\_matrix);

slope = (nrho - 1)/(rho(end) - rho(1));

rho\_bin\_index = round(slope\*(rho\_matrix - rho(1)) + 1);

theta\_bin\_index = repmat(1:ntheta, size(x\_matrix, 1), 1);

% Take advantage of the fact that the SPARSE function, which

% constructs a sparse matrix, accumulates values when input

% indices are repeated. That¡¯s the behavior we want for the

% Hough transform. We want the output to be a full (nonsparse)

% matrix, however, so we call function FULL on the output of

% SPARSE.

h = h + full(sparse(rho\_bin\_index(:), theta\_bin\_index(:), ...

val\_matrix(:), nrho, ntheta));

end

1. **Houghpeaks.m**

function [r, c, hnew] = houghpeaks(h, numpeaks, threshold, nhood)

%HOUGHPEAKS Detect peaks in Hough transform.

% [R, C, HNEW] = HOUGHPEAKS(H, NUMPEAKS, THRESHOLD, NHOOD) detects

% peaks in the Hough transform matrix H. NUMPEAKS specifies the

% maximum number of peak locations to look for. Values of H below

% THRESHOLD will not be considered to be peaks. NHOOD is a

% two-element vector specifying the size of the suppression

% neighborhood. This is the neighborhood around each peak that is

% set to zero after the peak is identified. The elements of NHOOD

% must be positive, odd integers. R and C are the row and column

% coordinates of the identified peaks. HNEW is the Hough transform

% with peak neighborhood suppressed.

%

% If NHOOD is omitted, it defaults to the smallest odd values >=

% size(H)/50. If THRESHOLD is omitted, it defaults to

% 0.5\*max(H(:)). If NUMPEAKS is omitted, it defaults to 1.

% Copyright 2002-2004 R. C. Gonzalez, R. E. Woods, & S. L. Eddins

% Digital Image Processing Using MATLAB, Prentice-Hall, 2004

% $Revision: 1.5 $ $Date: 2003/11/21 13:34:50 $

if nargin < 4

nhood = size(h)/50;

% Make sure the neighborhood size is odd.

nhood = max(2\*ceil(nhood/2) + 1, 1);

end

if nargin < 3

threshold = 0.5 \* max(h(:));

end

if nargin < 2

numpeaks = 1;

end

done = false;

hnew = h; r = []; c = [];

while ~done

[p, q] = find(hnew == max(hnew(:)));

p = p(1); q = q(1);

if hnew(p, q) >= threshold

r(end + 1) = p; c(end + 1) = q;

% Suppress this maximum and its close neighbors.

p1 = p - (nhood(1) - 1)/2; p2 = p + (nhood(1) - 1)/2;

q1 = q - (nhood(2) - 1)/2; q2 = q + (nhood(2) - 1)/2;

[pp, qq] = ndgrid(p1:p2,q1:q2);

pp = pp(:); qq = qq(:);

% Throw away neighbor coordinates that are out of bounds in

% the rho direction.

badrho = find((pp < 1) | (pp > size(h, 1)));

pp(badrho) = []; qq(badrho) = [];

% For coordinates that are out of bounds in the theta

% direction, we want to consider that H is antisymmetric

% along the rho axis for theta = +/- 90 degrees.

theta\_too\_low = find(qq < 1);

qq(theta\_too\_low) = size(h, 2) + qq(theta\_too\_low);

pp(theta\_too\_low) = size(h, 1) - pp(theta\_too\_low) + 1;

theta\_too\_high = find(qq > size(h, 2));

qq(theta\_too\_high) = qq(theta\_too\_high) - size(h, 2);

pp(theta\_too\_high) = size(h, 1) - pp(theta\_too\_high) + 1;

% Convert to linear indices to zero out all the values.

hnew(sub2ind(size(hnew), pp, qq)) = 0;

done = length(r) == numpeaks;

else

done = true;

end

end

1. **Houghpixels.m**

function [r, c] = houghpixels(f, theta, rho, rbin, cbin)

%HOUGHPIXELS Compute image pixels belonging to Hough transform bin.

% [R, C] = HOUGHPIXELS(F, THETA, RHO, RBIN, CBIN) computes the

% row-column indices (R, C) for nonzero pixels in image F that map

% to a particular Hough transform bin, (RBIN, CBIN). RBIN and CBIN

% are scalars indicating the row-column bin location in the Hough

% transform matrix returned by function HOUGH. THETA and RHO are

% the second and third output arguments from the HOUGH function.

% Copyright 2002-2004 R. C. Gonzalez, R. E. Woods, & S. L. Eddins

% Digital Image Processing Using MATLAB, Prentice-Hall, 2004

% $Revision: 1.4 $ $Date: 2003/10/26 22:35:03 $

[x, y, val] = find(f);

x = x - 1; y = y - 1;

theta\_c = theta(cbin) \* pi / 180;

rho\_xy = x\*cos(theta\_c) + y\*sin(theta\_c);

nrho = length(rho);

slope = (nrho - 1)/(rho(end) - rho(1));

rho\_bin\_index = round(slope\*(rho\_xy - rho(1)) + 1);

idx = find(rho\_bin\_index == rbin);

r = x(idx) + 1; c = y(idx) + 1;

1. **Houghlines.m**

function lines = houghlines(f,theta,rho,rr,cc,fillgap,minlength)

%HOUGHLINES Extract line segments based on the Hough transform.

% LINES = HOUGHLINES(F, THETA, RHO, RR, CC, FILLGAP, MINLENGTH)

% extracts line segments in the image F associated with particular

% bins in a Hough transform. THETA and RHO are vectors returned by

% function HOUGH. Vectors RR and CC specify the rows and columns

% of the Hough transform bins to use in searching for line

% segments. If HOUGHLINES finds two line segments associated with

% the same Hough transform bin that are separated by less than

% FILLGAP pixels, HOUGHLINES merges them into a single line

% segment. FILLGAP defaults to 20 if omitted. Merged line

% segments less than MINLENGTH pixels long are discarded.

% MINLENGTH defaults to 40 if omitted.

%

% LINES is a structure array whose length equals the number of

% merged line segments found. Each element of the structure array

% has these fields:

%

% point1 End-point of the line segment; two-element vector

% point2 End-point of the line segment; two-element vector

% length Distance between point1 and point2

% theta Angle (in degrees) of the Hough transform bin

% rho Rho-axis position of the Hough transform bin

% Copyright 2002-2004 R. C. Gonzalez, R. E. Woods, & S. L. Eddins

% Digital Image Processing Using MATLAB, Prentice-Hall, 2004

% $Revision: 1.4 $ $Date: 2003/10/26 22:34:10 $

if nargin < 6

fillgap = 20;

end

if nargin < 7

minlength = 40;

end

numlines = 0; lines = struct;

for k = 1:length(rr)

rbin = rr(k); cbin = cc(k);

% Get all pixels associated with Hough transform cell.

[r, c] = houghpixels(f, theta, rho, rbin, cbin);

if isempty(r)

continue

end

% Rotate the pixel locations about (1,1) so that they lie

% approximately along a vertical line.

omega = (90 - theta(cbin)) \* pi / 180;

T = [cos(omega) sin(omega); -sin(omega) cos(omega)];

xy = [r - 1 c - 1] \* T;

x = sort(xy(:,1));

% Find the gaps larger than the threshold.

diff\_x = [diff(x); Inf];

idx = [0; find(diff\_x > fillgap)];

for p = 1:length(idx) - 1

x1 = x(idx(p) + 1); x2 = x(idx(p + 1));

linelength = x2 - x1;

if linelength >= minlength

point1 = [x1 rho(rbin)]; point2 = [x2 rho(rbin)];

% Rotate the end-point locations back to the original

% angle.

Tinv = inv(T);

point1 = point1 \* Tinv; point2 = point2 \* Tinv;

numlines = numlines + 1;

lines(numlines).point1 = point1 + 1;

lines(numlines).point2 = point2 + 1;

lines(numlines).length = linelength;

lines(numlines).theta = theta(cbin);

lines(numlines).rho = rho(rbin);

end

end

end

1. **Main Code**

%%% Robot Vision%%%

%%% Dept. of Electronic Engineering

%%% 201314651 Lee Wonjai

% read the targeted image

IM\_Edge = imread('C:\Users\user\OneDrive\¹ÙÅÁ È­¸é\2019 Æ¯º°, 4ÇÐ³â\4ÇÐ³â 2ÇÐ±â\·Îº¿ºñÀü\Homeworks\hw7\Edge.png');

[H,theta,rho] = hough(IM\_Edge);

figure, imshow(H, [], 'XData',theta, 'YData',rho,'InitialMagnification','fit')

axis on, axis normal

xlabel('\theta'), ylabel('\rho')

[r, c] = houghpeaks(H, 10);

hold on

plot(theta(c), rho(r), 'linestyle', 'none', 'marker', 's', 'color', 'w')

lines = houghlines(IM\_Edge, theta, rho, r, c);

figure, imshow(IM\_Edge), hold on

for k = 1:length(lines)

xy = [lines(k).point1; lines(k).point2];

plot(xy(:,2), xy(:,1), 'lineWidth', 4, 'Color', [.6 .6 .6]);

end

r1 = [];

c1 = [];

for i = 1:length(c)

if theta(c(i)) <= 10 && theta(c(i)) >= -10

r1(end+1) = r(i);

c1(end+1) = c(i);

end

end

figure, imshow(H, [], 'XData',theta, 'YData',rho,'InitialMagnification','fit')

axis on, axis normal

xlabel('\theta'), ylabel('\rho')

hold on

plot(theta(c1), rho(r1), 'linestyle', 'none', 'marker', 's', 'color', 'w')

lines1 = houghlines(IM\_Edge, theta, rho, r1, c1);

figure, imshow(IM\_Edge), hold on

for j = 1:length(lines1)

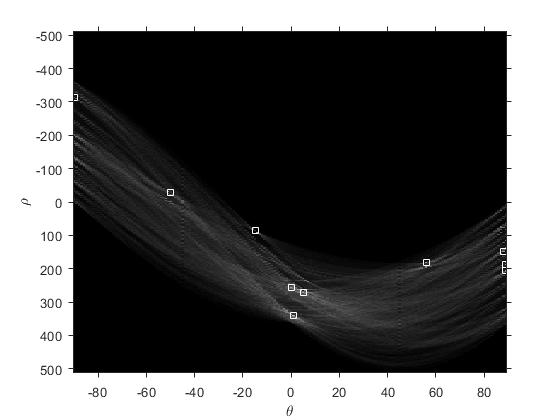
xy1 = [lines1(j).point1; lines1(j).point2];

plot(xy1(:,2), xy1(:,1), 'lineWidth', 4, 'Color', [.6 .6 .6]);

end

**02 Result 1**

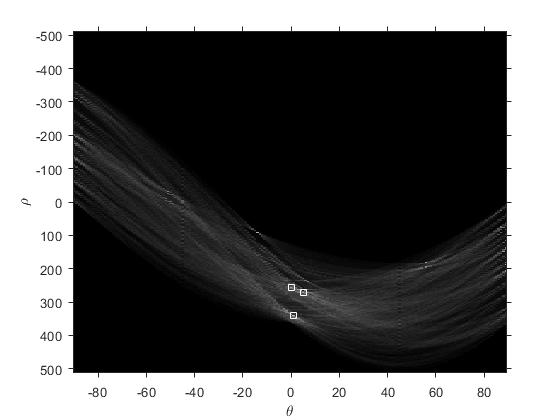
1. **Hough Transformed Function and 10 Peaks**



1. **10 peaks and Image**



1. **Hough Transformed Function within +- 10 Degree peaks**



1. **Within +- 10 Degree Peaks Image**

