HOUGH TRANSFORM TO IDENTIFY THE EQUATIONS OF THE EDGE

Ayush Aggarwal (11177)

IIT Kanpur, Department of Civil Engineering, Geo-Informatics Division, India- ayushagg@iitk.ac.in

HW 7, Course-CE676A

KEY WORDS: Hough transform, Edge detection

ABSTRACT:

In this lab exercise we have been provided with a sample set of points in the file 'CE676_HA7.txt', which contains xyz triplets. These points belong to the edges of a planar feature (e.g. rooftop, signboard, or any other feature). We have to develop a code that uses "Hough Transform" to identify the equations of the edge.

1. INTRODUCTION

1.1 Lidar

The popularity of LIDAR in Photogrammetry and Remote Sensing is constantly increasing as it allows for direct acquisition of three-dimensional (3D) dense information of the earth surface. Extremely dense 3D recordings of point clouds and a very reliably technology are highlights of LIDA.

Lidar uses ultraviolet, visible, or near infrared light to image objects. It can target a wide range of materials, including non-metallic objects, rocks, rain, chemical compounds, aerosols, clouds and even single molecules.

1.2 Las File

The LAS file is contain LIDAR point data cloud. Different LIDAR hardware and software tools output data in this common open format. The data is generally be put into this format from software which combines GPS, IMU, and laser pulse range data to produce X, Y, and Z point data. This format is binary consisting of a header, Variable Length Records, and point data.

1.3 Visualisation

Visualisation can be achieved by converting the point cloud to various types of raster images, but it is also possible to browse through 3D point clouds and visualise them using point-based rendering techniques like point splatting. Airborne laser scanning data or data from a single terrestrial scan can, however, be considered as 2.5D and be converted to height or range imagery.

The creation of a height image from a point cloud consists of three steps:

1) definition of the image grid;
2) determination of the height for each pixel
3) Conversion of the height to a grey value or color.

1.4 Hough Transform

The Hough transform is a technique which can be used to isolate features of a particular shape within an image. Because it requires that the desired features be specified in some parametric form, the classical Hough transform is most commonly used for the detection of regular curves such as lines, circles, ellipses, etc. A generalized Hough transform can be employed in applications where a simple analytic description of a feature(s) is not possible. Due to the computational complexity of the generalized Hough algorithm, we restrict the main focus of this discussion to the classical Hough transform. Despite its domain restrictions, the classical Hough transform (hereafter referred to without the classical prefix) retains many applications, as most manufactured parts (and many anatomical parts investigated in medical imagery) contain feature boundaries which can be described by regular curves. The main advantage of the Hough transform technique is that it is tolerant of gaps in feature boundary descriptions and is relatively unaffected by image noise.

How it works

This technique is used to detect the straight lines like building contour polygons, and curves such as circles and ellipses. With the 3D point cloud, the demand is increased for detecting 3D planes. In this context, the 2D Hough-transform has been extended to 3D

The principle of the 2D Hough-transform is the representation of a points set, defined initially in the Euclidian space, in another space. This transform allows detecting the points composing specific geometric primitives. For example, in (OXY) space, the equation of a line has the form (1).

$$\mathbf{Y = a.\, X + b} \qquad \mathbf{(1)}$$

where (a, b) are the line parameters. This line can be represented by a point with coordinates (a, b) in the parameter space (O' a b). In an opposite way, one point (Xi, Yi) belonging to the space (OXY) is represented by a line in the parameter space (O' a b) as expressed in Equation 2.

$$\mathbf{b = - Xi\, .a + Yi} \qquad \mathbf{(2)}$$

where (Xi, Yi) are the parameters of this line. Supposing that M1, M2, … Mn are a set of points in the space (OXY) and that they belong to the line P following Equation 1. Each one of these points represents a line in the parameter space. The intersection of these lines in the parameter space is the point (a1, b1) which represents the parameters of the line P in a 2D-space.

If the line equation has the form X = constant, then it can not be presented in the parameter space (O' a b), because the Y-axis coefficient is equal to zero. In order to solve this problem, it is suggested to use the normal form of the line (Equation 3).

$$\cos\theta .\ X + \sin\theta .\ Y = \rho \qquad (3)$$

where θ and ρ are the parameters of the normal passing through the origin (see Fig.1).

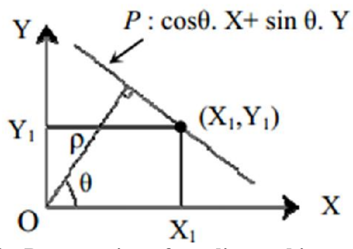


Fig 1 - Presentation of one line and its normal in a 2D-space

The same principle can be applied in a 3D case in considering that one plane belonging to the (OXYZ) space (Equation 4) can be represented by a point (a, b, c) in the parameter space (O'abc).

$$Z = a.\ X + b.\ Y + c \qquad (4)$$

In the same manner, if the plane equation has the form (5), then it can not be presented in the parameter space because the Zaxis coefficient is equal to zero. In order to solve this problem, (Overby et al., 2004) suggest to use also the normal form of the plane (Equation 6).

$$a X + b Y + c = 0 \qquad (5)$$

$$\cos\theta .\ \cos\varphi .\ X + \sin\theta .\ \cos\varphi .\ Y + \sin\varphi .\ Z = \rho \qquad (6)$$

where θ, φand ρare the parameters of the plane normal passing through the origin (see Fig.2).

So, θ, φand ρare constant and the parameter space is (O' θ φ ρ). In this case, one point (X1, Y1, Z1) in the 3D-space represents a sinusoidal surface in the parameter space. Since the principles of the 3D Hough-transform are explained, the aim of the next section is to deliver its algorithm

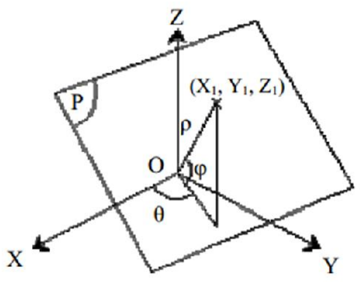


Fig 2 - Representation of plane equation elements in the normal form

2. SOFTWARES/DATA USED

1) Matlab
2) LAS_Data_for_LASViewer
3) CE676_HA7.txt

3. METHODOLOGY

Firstly import the file having coordinates then follow the algorithm given –The input data are the steps on θ, φand ρ axis (discrete intervals), called θ_step, φ_step and ρ_steprespectively. The 3D point cloud is represented by three coordinate lists X, Y and Z. Algorithm 1 presents the pseudo code of the 3D Hough transform.

Algorithm 1: 3D Hough-transform for plane detection

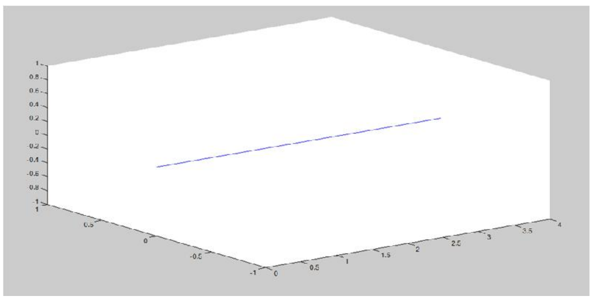
```
1. X_min = min(X); Y_min = min(Y); Z_min = min (Z)
2. X_max = max(X); Y_max = max(Y); Z_max = max (Z)
3. Calculation of: Dis_min; Dis_max
4. θ = from 0 to 360, step = θ_step; n_θ = length(θ)
5. φ = from -90 to +90, step = φ_step; n_φ = length(φ)
6. n_ρ = 2* (Dis_max - Dis_min) / ρ_step
7. ρ = from Dis_min to Dis_max; step = ρ _step
8. θ_mat (n_φ, n_θ) = [θ θ θ … θ]' *π/180
9. φ_mat(n_φ, n_θ) = [φ φ φ …. φ] * π /180
10. H(n_θ, n_φ, n_ρ) = 0
11. ratio = (n_ρ – 1)/( ρ (n_ρ) – ρ (1))
12. for k = 1 to length(X)
13. ρ_mat = cos (φ_mat)*cos (θ_mat)* X (k) + ...
       cos (φ_mat)*sin (θ_mat)* Y (k)+ sin(φ_mat)* Z (k)
14. ρ_indix = round (ratio *( ρ_mat – ρ (1)+1))
15.  for i = 1 to n_φ
16.  for j = 1 to n_ θ
17.  H (j, i, ρ_index (i, j)) = H (j, i, ρ_index (i, j)) +1
18. next j ; next i ; next k
```

In this algorithm, Dis_minand Dis_maxare the distances between the origin and the two extremities of the cloud points calculated at lines 1 and 2; His a 3D matrix; θ_mat, φ_matand ρ_mat are 2D matrices; θ, φand ρare three lists.

The result of the algorithm is the 3D matrix Hwhich contains the representation of the original cloud in the parameter space. Each point of (OXYZ) space gives a sinusoidal surface in the parameter space.

4. RESULTS

Hough room line plot are shown below –



5. CONCLUSION

We learned mat lab for getting plane and lines from Hough Transform.

6. REFERENCES

[1] Lohani, B., R. K. Mishra, 2007. Generating LiDAR data in laboratory: LiDAR simulator. International Archive of Photogrammetry and Remote Sensing, vol. XXXVI (3).

[4] BARINOVA O., LEMPITSKY V. S., KOHLI P.: On detection of multiple object instances using Hough transforms. In CVPR (2010), IEEE, pp. 2233–2240.

All Accessed on 26/4/2015

7. CODE

```
% Ayush Aggarwal
% 11177
% Predicting Planes using hough
transform
clc;
clear all;
D =
textread('CE676_HA7.txt','','delimiter
',',' );
X=D(:,1);
Y=D(:,2);
Z=D(:,3);
leng=length(D(:,1));

%%
% variables controlling plane search
dist_incr = 0.03;
theta_incr = 2;
phi_incr=2;
maxx=max(D(:, 1));
minx=min(D(:, 1));
maxy=max(D(:, 2));
miny=min(D(:, 2));
maxz=max(D(:, 3));
minz=min(D(:, 3));
Dis_min= sqrt(minx^2+miny^2+minz^2);
Dis_max= sqrt(maxx^2+maxy^2+maxz^2);


%Setting up Houghroom matrix based on
size of the D set
% we are taking 3d variable as
distance from origin theta and phi
distance = Dis_min: dist_incr :
Dis_max;
theta = 0 : theta_incr : 90-
theta_incr;
phi= 0:phi_incr:90-phi_incr;   % as x
y z are positive
theta_rad = theta*pi/180;
phi_rad=phi*pi/180;
c_phi=cos(phi_rad);
t_theta= length(theta);
t_phi=length(phi);
t_distance=length(distance);
ratio= (t_distance-
1)/(distance(t_distance)-distance(1));
Hough_room = zeros(length(theta),
length(phi),length(distance)); %
making matrix for putting hough values

%%
% if parameters lie with in various
parameter then increse the hough room
count
for l=1:1:t_distance
   for k=1:1:t_phi
     for j=1:1:t_theta
        for i = 1:leng
            dist =
X(i)*(cos(phi_rad(k))*cos(theta_rad(j)
)) +
Y(i)*(cos(phi_rad(k)).*sin(theta_rad(j
)))+ Z(i)*(sin(phi_rad(k))) ;
            % taking error in distamce
= error in point given in question for
less
            % computation so hough
interval decided by it
            if
distance(l)<(dist+0.015) &
distance(l)> (dist-0.015)
                 Hough_room(k, j, l) =
Hough_room(k, j, l) +1;
            end
        end
     end
   end
end

%% Plot hough room
plot3(Hough_room(:,1),
Hough_room(:,2), Hough_room(:,3));

%%
%For hough room thresolding
thresh = 0.5
*(max(max(Hough_room(:))));

% getting points of result by hough
room thershold
[r_theta r_phi r_distance] =
find(Hough_room > thresh);

r_Houghroom = Hough_room - thresh;
r_hough_dist = [];
r_hough_theta = [];
r_hough_phi=[];

%Finding x, y, z of the cloud in
result Hough_room after thershold.
count=0;
```

```
for l = 1:t_distance
    for j=1:t_theta
        for k=1:t_phi
        if r_Houghroom(j, k, l) >= 0
            count=count+1;
            r_hough_dist(count,1) =
distance(l);
            r_hough_theta(count,1) =
theta(j);
            r_hough_phi(count,1) =
phi(k);
        end
        end
    end
end

% Calculation of lines.
r_hough_dist = r_hough_dist *
dist_incr;
r_hough_theta = (r_hough_theta *
theta_incr) - theta_incr;
r_hough_phi = (r_hough_phi*phi_incr)-
phi_incr;
line=[r_hough_dist r_hough_theta
r_hough_phi];
dlmwrite('result_line.txt',line,'delim
iter',' ','precision','%.6f');
```