Chapter

Laser Point Cloud Segmentation in MATLAB

Bahadır Ergün and Cumhur Şahin

Abstract

Currently, as a result of the massive continuous advancements in laser measurement technology, possibilities of map production are broadened, the loss of time and the waste of material sources are highly prevented, and the accuracy and precision of the obtained results are significantly improved. In the view of engineering concept. However, big data which are from laser point clouds have been especially used in the significant procedures of surveying studies. Programming methods are dependent in each studies. In the necessity of the applications, the coding procedure has more efficient, the data of work has increased, and time has been consumed. The coding methods have necessarily been optimized for working together especially in the big data studies. In this section, an automated survey (building facade surveying) is produced from scanning data by means of coding in MatLAB.

Keywords: surveying, laser point cloud data, segmentation, object determination, coding in MatLAB

1. Introduction

Nowadays, laser scanning and modeling technology have been extensively used in city documentation and cultural heritage studies besides the technique of imaging for global representation in the internet survey and navigation applications. In the view of engineering concept. However, big data which are from laser point clouds have been especially used in the significant procedures of surveying studies. Programming methods are dependent in each studies. In the necessity of the applications, the coding procedure has more efficient, the data of work has increased, and time has been consumed. The coding process of big data process must be modeled in the hardware systems requires receiving consideration. This process has been related to hardware construction by electronic and computer engineering vision. This process has mathematical model and algorithm, which has been concerned surveying and computer programming engineering vision. In this chapter, Matlab coding models including functional and stochastic properties have been suggested and discussed for operational process within laser scanning data segmentation in surveying studies.

2. Process

2.1 Data structure

Laser scanners scan the object in horizontal and vertical directions under a certain angle as a series of points and this allows the object to be displayed as a point cloud. In order to determine each of the laser points, measurement of scanner-centric polar coordinates are made. The measured points are slanted distance to point P, the angle between X-axis and horizontal plane α, and the angle of inclination of the horizontal plane measuring line φ. As illustrated in **Figure 1**, the initial point of terrestrial laser scanners are considered to be the positioned points. These measurements are based entirely on their local coordinate systems.

The resulting point cloud data is processed in formats related to the coordinate and the angle. The processing is carried out as follows: DXF for CAD models, ASCII for surface modeling, VRML format for visualization, and txt or pts. Software which varies with different laser scanner instrumentation can be used to obtain the point cloud data.

Laser scanners initially obtain the X,Y,Z Cartesian coordinates inside a second coordinate system which is located at the center of the station point and then they scan the surface of the object. In addition to the three-dimensional coordinates, the resulting data includes the density of the returning signal in terms of RGB (Red, Green, Blue) depending on the structure of the surface in question and the distance of measurement. Modeling of the scanned object and environment gets easier with recorded RGB density values. The dense data obtained by scanning is called a point cloud. **Table 1** displays the formation of txt data linked to the point cloud data.

There is a software in target-oriented modules to obtain raw data, to convert data to a workable format, and to perform the texturing process (if necessary) etc.

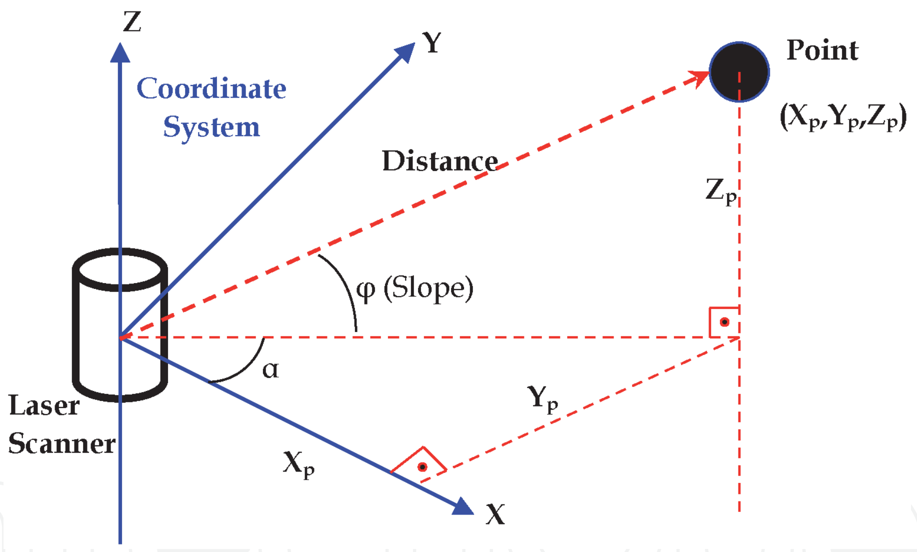

Figure 1.
TLS local coordinate system.

Geometric Information			Radiometric Information			
X (m)	Y (m)	Z (m)	Gamma	Red	Green	Blue
0.153229	0.521369	−0.004161	−76	91	115	113
0.270996	0.521319	−0.004880	−75	87	109	107
0.153229	0.467538	−0.009394	41	75	94	98
0.270874	0.467157	−0.006026	−167	81	100	97
0.216461	0.494419	−0.006889	−170	79	98	96

Table 1.
Point cloud data formation in ASCII format.

Figure 2 shows the point cloud data that is represented with RGB density values. A Leica HDS – 3000 terrestrial laser scanner is used to scan the point cloud data.

2.2 Programming flow procedure

During the documentation of the coordinate information of laser scanner point cloud data, there is no regular data order and classification [1]. For segmentation, points with known three-dimensional coordinates must be selected from all point cloud data. The algorithm is formalized with mathematical surface or point clustering techniques. Planar surfaces or points including depth parameters can be extracted by using various methods. In addition, the surface points of the assigned surfaces are filtered. Once the building's planar surfaces are obtained, various methods can be used to extract property boundaries.

Programming flow procedure could be designed in this method in Matlab programming in **Figure 3**.

Deciding segmentation method is a dynamic process in Matlab programming flow. There are three different segmentation methods have been used in this step.

Geometric segmentation is based on geometric information of point cloud data. Radiometric segmentation is based on radiometric information of point cloud data. Hybrid segmentation is based on all information of point cloud data. The mathematical model of these segmentation methods could be based on not only

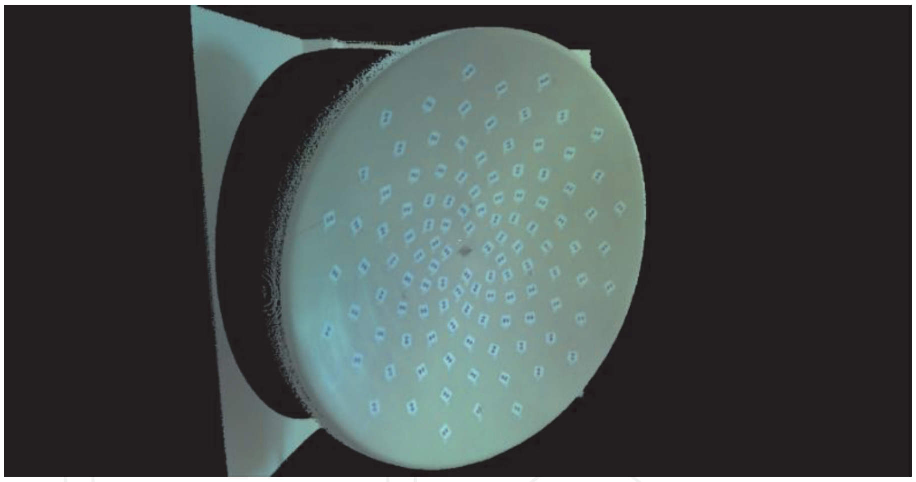

Figure 2.
Cyclone 5.2 point cloud image.

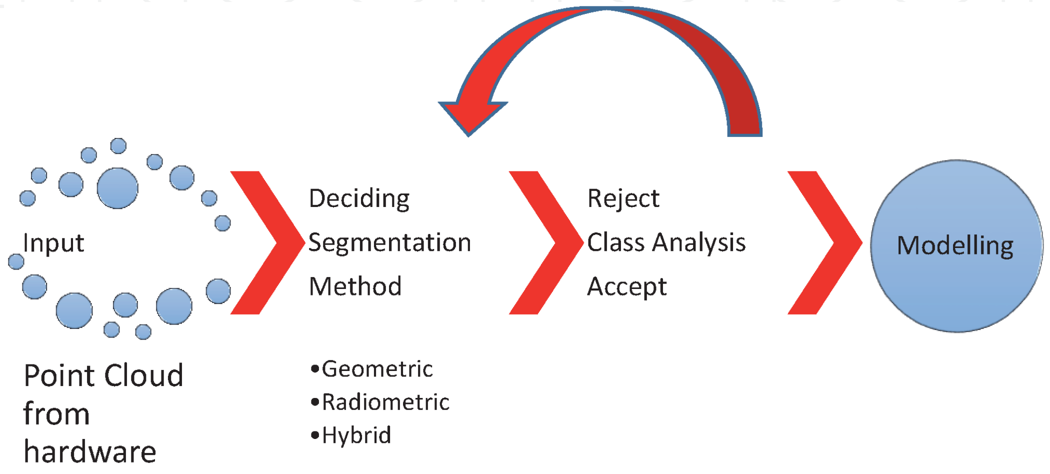

Figure 3.
Programming flow procedure.

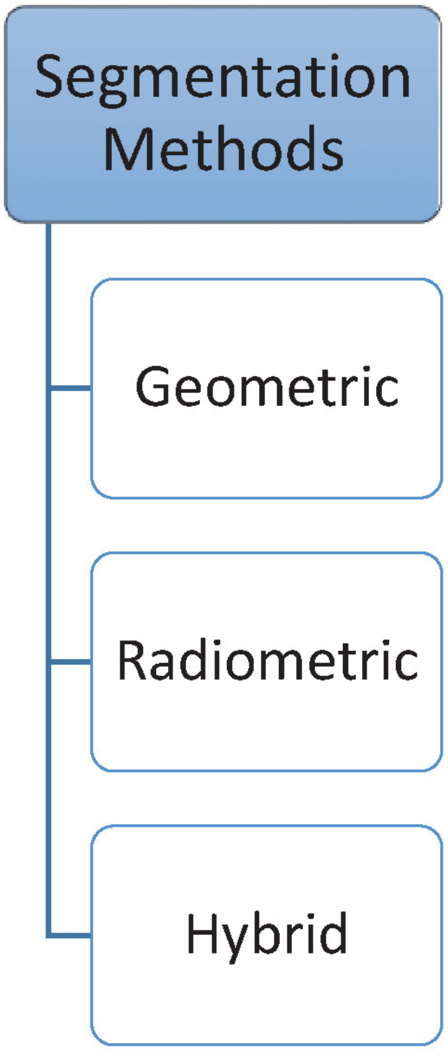

Figure 4.
Segmentation methods.

conventional methods but also expert systems (Fuzzy systems, SVM, etc.) as shown in **Figure 4**.

3. Examples of geometric segmentation in Matlab

3.1 Point segmentation

Algorithm of this study which aims at filtering laser point cloud data of parallel surfaces in indoor areas by the help of filtering function is shown in **Figure 5** [2].

Selecting one of the parallel indoor surfaces as the reference plane by an operator is the first step of filtering function algorithm.

Distinct surfaces define the point cloud data which is illustrated in **Figure 6**. Planar surfaces like walls generally define the indoor areas. **Figure 6** displays point cloud data and various surface structures in a three dimensional coordinate system.

Mathematical function that represents plane surface is given in Eq. (1).

$$Ax + By + Cz + D = 0 \tag{1}$$

Eq. (1) shows that surface function consists of 4 parameters: A, B, C, D. The selected reference plane surface can be expressed mathematically by calculating these four parameters. An operator must read and manually enter the selected point

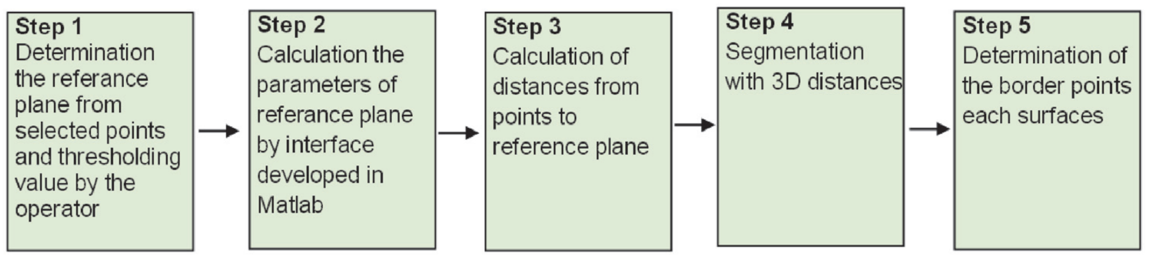
Step 1
Determination the referance plane from selected points and thresholding value by the operator
Step 2
Calculation the parameters of referance plane by interface developed in Matlab
Step 3
Calculation of distances from points to reference plane
Step 4
Segmentation with 3D distances
Step 5
Determination of the border points each surfaces

Figure 5.
Geometric segmentation steps.

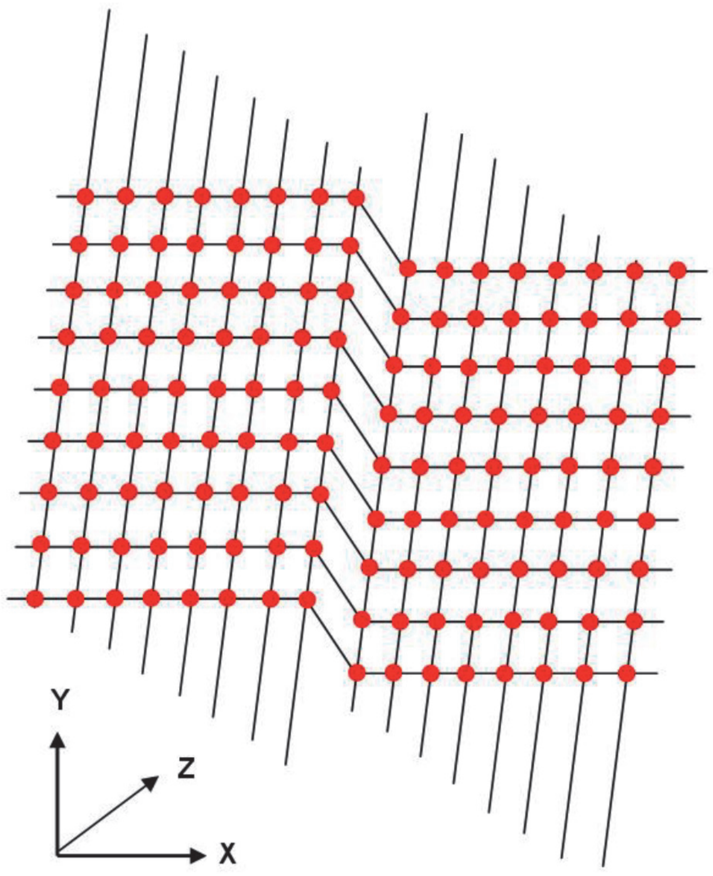

Figure 6.
Point cloud data structure in 3D coordinate system.

coordinates of reference plane. The parameters of selected reference plane surface are determined by this way. A plane can be mathematically defined with reference to four parameters. So, we can only define the parameters of reference plane with four points. An operator selects more than four points in the same reference plane and determines the parameters of reference plane in adjustment process. Operator manually enters the threshold value secondly. Threshold value can be defined as the minimum difference of depth during the filter operation. All the operations are performed automatically by a Matlab-based software, other than the two stages of the algorithm.

Calculating the parameters of the selected reference plane is the second step of geometric segmentation algorithm. Once the operator manually chooses multiple (five or more) points as part of the first step of algorithm, the Matlab-based interface automatically determines four parameters which represent the reference plane in the adjustment computation. Thus, the adjusted reference plane is created in this step.

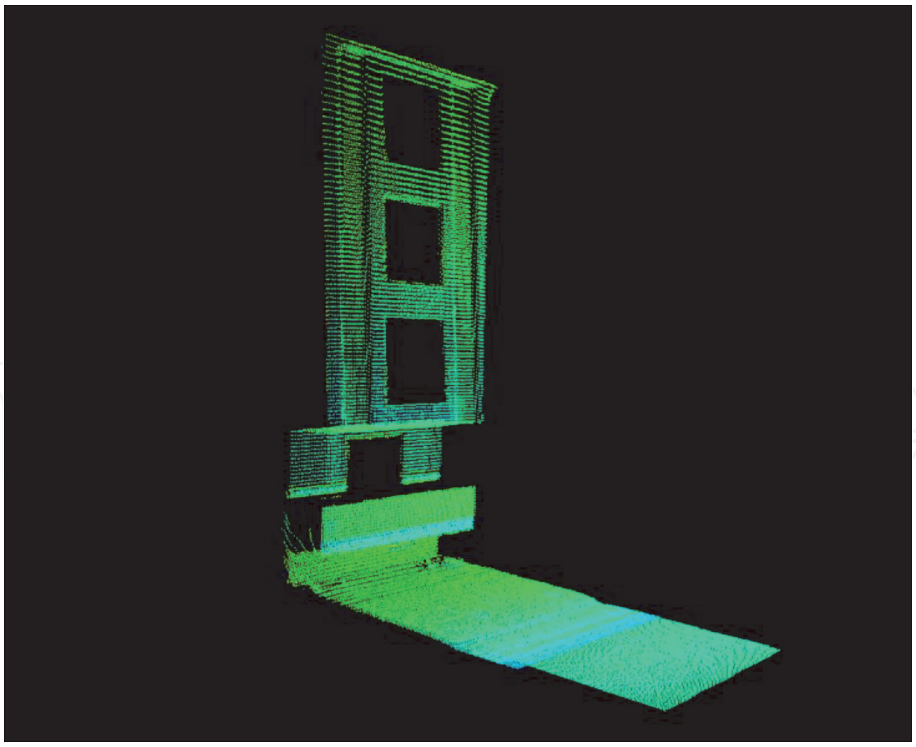

Figure 17.
Mobil point cloud data of application Array.

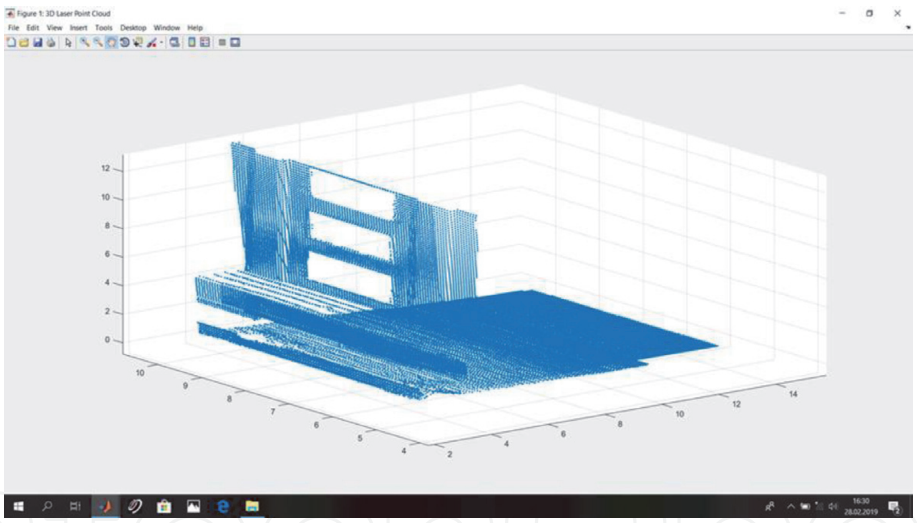

Figure 18.
Three-dimensional laser point cloud data in Matlab.

and minimum X and Y values from the ground points. When determining the center axis of the road, the axis was used on the points which were taken as $Z_{1=} Z$ and had a depth of Z_1 and above.

These processes were calculated with the following Eqs. (4) and (5):

$$Axis_y = Ground_{y_{min}} + \left(\frac{Ground_{y_{max}} - Ground_{y_{min}}}{2}\right) \tag{4}$$

$$Axis_x = Ground_{x_{min}} + \left(\frac{Ground_{x_{max}} - Ground_{x_{min}}}{2}\right) \tag{5}$$

Other than the values which were assigned to the class column with the value of 0 (ground point), the points assigned with class value 1 (building points) were saved as Matrix_Building_seg matrix and its graph was plotted in the **Figure 19**.

The building points were separated from the ground height of 35 cm, but a new classification was made on the basis of the above-mentioned road center axis to determine whether the building was on the left or right side of the road axis. For this purpose, a value of 1 was assigned to points smaller than $Axis_x$ (depth of the road axis) and a value of 2 was assigned to points greater than $Axis_x$. The points assigned with the value of 1 are shown in the Matrix_Building_Left matrix, and the points assigned with the value of 2 are shown in the Matrix_Building_Right matrix.

It is assumed that the building lays on the side with the high number of points. So, the number of points of the Matrix_Building_Left and Matrix_Building_Right matrices were calculated and the values of the Matrix_Building_Left matrix were written as Matrix_Building_Segment matrix where the building points were completely separated and then, the building points were plotted. Also, if there were no points exceeding the height limit (Z_1) on the right side of the road center axis, although this data was not included in our data, this classification step would have to be made in order to completely make a distinction of the building points in the **Figure 19** also.

After determining the building points, building points were transferred into a reference surface to draw the building facade survey. A mathematical filtering function was applied in this stage.

The segmentation works on the basis of surface-dependent height differences within the maximums and minimums of the surface function that forms the mathematical model. Surface equation is expressed as Eq. (6):

$$ax_n + by_n + cz_n + d = 0 \tag{6}$$

where n is the number of points.

An exponential filtering method is used as the filtering function. Filtering function is determined as Eq. (7):

$$\mathrm{f}(\mathrm{x}) = e^{\Delta Building} = \frac{1}{e^{\Delta Depth}} \tag{7}$$

Where $\Delta X_{Building}$ and Δ_{Depth} were calculated with the following Eqs. (8) and (9):

$$\Delta X_{Building} = \frac{d + bZ_{Building} + aY_{Building}}{c} \tag{8}$$

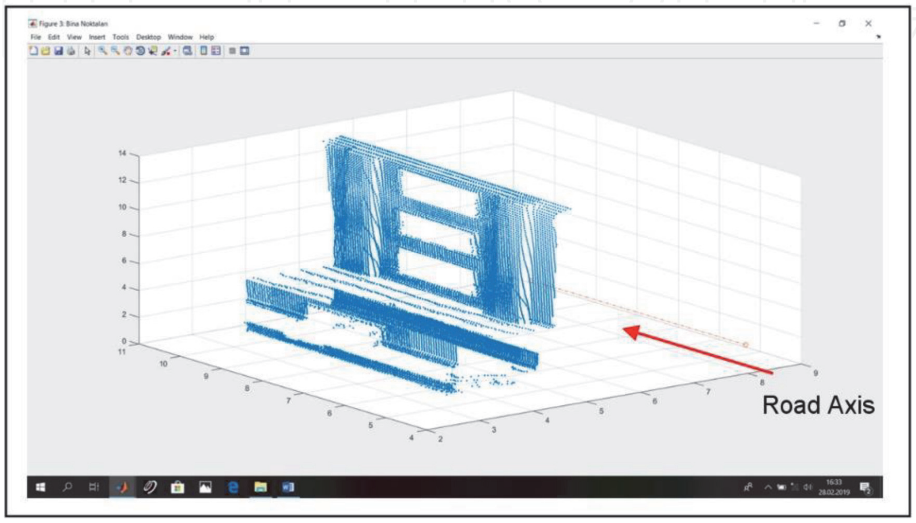

Figure 19.
Road center Axis and point cloud data without point cloud of road in Matlab.

$$\Delta_{Depth} = X_{Building} - X_{max} \tag{9}$$

a, b, c, d represent plane equation parameters, $\Delta X_{Building}$ is the depth of building points to the reference surface, Δ_{Depth} is depth differences of building points to the reference surface, $e^{\Delta Bina}$ is result values of the function, $e^{\Delta Depth}$ is exponential values of differences of depth, and $Eksen_y$ represents the y value of road center axis.

For the segmentation process, the building reference surface points were determined at first. Then, the limits of the reference surface were resolved by using the minimum and maximum points of Z (height) and Y (length) of the building points and a rectangular surface was created. Only the maximum value of X (depth) was used. This is due to the fact that the bottom window (Window 1) of the building, which we were to transfer to the surface is located more on the backside than the other three windows. Also, when the minimum value of X is taken, there is a possibility that the points behind the bottom window with the smaller depth values might interfere with the bottom window and building points. The specified reference surface points are written into a Building_Reference_Surface_Points.txt file in the **Table 2**. The Surface graphic has shown in the **Figure 20**.

After the reference surface points are determined, the reference plane parameters of the building are calculated using these points. The reference surface is drawing in Matlab has shown in the **Figure 21**, then the PRight matrix is created and plotted. It is written into a Building_Reference_Plane_Par.txt file.

X (m.)	Y (m.)	Z (m.)
5.424	10.537	13.598
5.424	4.762	13.598
5.424	10.537	1.014
5.424	4.762	1.014

Table 2.
Building reference surface points.

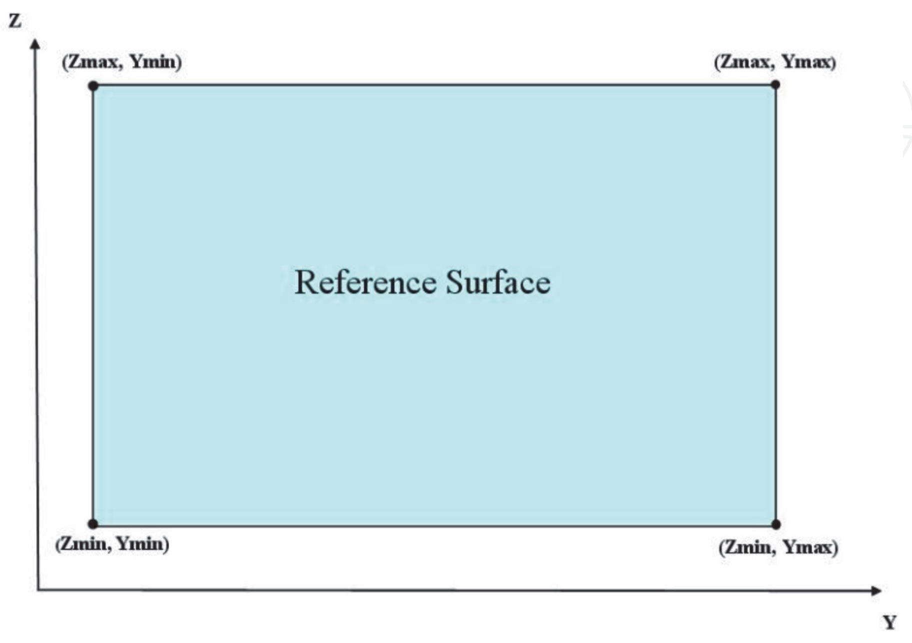

Figure 20.
Building reference surface points.

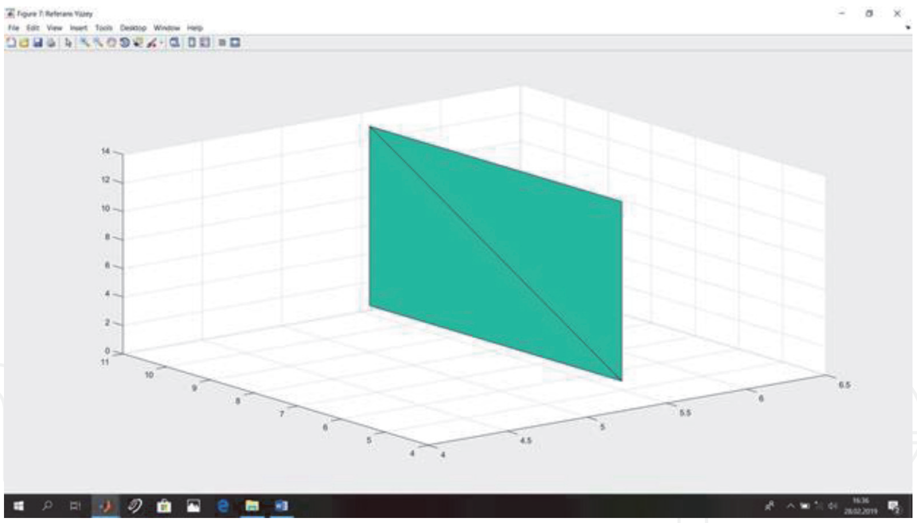

Figure 21.
Reference surface.

The following Eqs. (10–13) are used in the process of adjustment:

$$I = [4, 1] \tag{10}$$

$$N = BY^T BY \tag{11}$$

$$n = N^{-1}\, I \tag{12}$$

$$X = N^{-1}[I] \tag{13}$$

It represents unit matrix, BY is the matrix of detected maximum and minimum surface points, N is the normal equation coefficient matrix, n is the plain terms vector, and X represents the matrix of unknown values (surface parameters) in the **Table 3**.

Using the calculated parameters and the result values of the function ($e^{\Delta Building}$), Z and Y values of the building points are transferred to the reference plane. The X depth value ($Surface_{Depth}$) of the reference plane is calculated by averaging the maximum and minimum X values, and a $Surface_{Filter}$ matrix with the depth of the number of building points is created using the unit matrix. These processes are performed with the following Eqs. (14) and (15):

$$Surface_{Depth} = \frac{Surface_{X_{max}+} Surface_{X_{min}}}{2} \tag{14}$$

$$Surface_{Filter} = Surface_{Depth}.I_{Unit} \tag{15}$$

To use the coordinate values obtained by mathematical filtering in the Geometric.m code, the new point cloud matrix was created and written to the

Plane Parameter Coefficients	
a	−0.0034287
b	−0.018451
c	0.039046
d	0.0071988

Table 3.
Reference plane parameters.

Geometric_Filter_Input.txt file for the references surface.The distance between points was determined as 25 cm, Geometric.m code was executed and 180 new points or openings were written and graphed by creating opening_output_txt file. As a result of this process, the window openings were filled vertically. This segmentation points have shown in the **Figure 22**.

Since the gaps were calculated in blocks with gap filtering in Matlab, a reclassification was performed to determine which window the dots belonged to. A function which is similar to the exponential filter function used in the transfer of building points to the reference plane was used [16]. The windows which were drawn by using the coordinate values obtained from the result of point segmentation (distance between points is 14.5 cm) in red, surface segmentation in green, and conventional segmentation in magenta are shown in **Figure 23** and **Figure 24**.

To assign these points, the OP_Pen matrix with the number of lines (same as the building point numbers) and with a NaN value was created. Then, points within the 120 cm border were selected from the Matlab code matrix and printed on the OP_Pen matrix. In order to distinguish these assigned values from NaN values, the snip.m code was run once more. The points to be optimized for each window were differentiated and the files OP_PP1.txt, OP_PP2.txt, 0P_PP3.txt, OP_PP4.txt, which

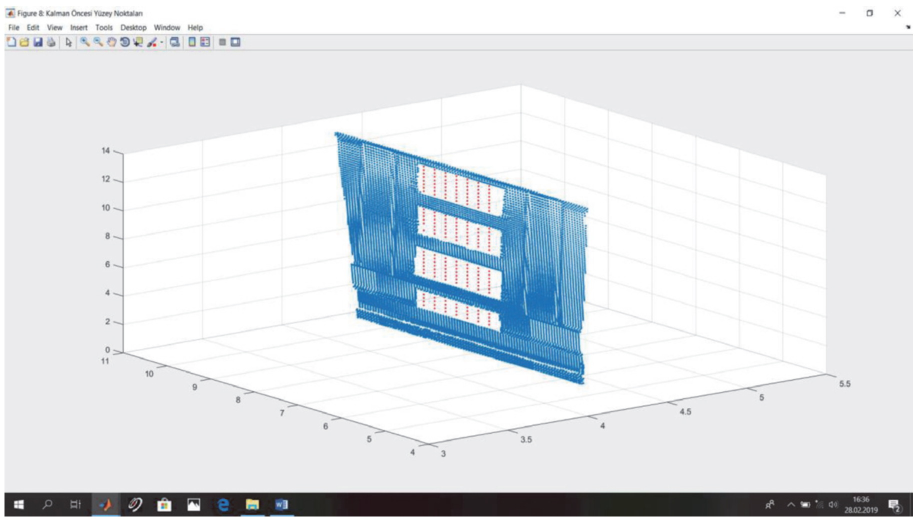
Figure 8: Kalman Öncesi Yüzey Noktaları
File Edit View Insert Tools Desktop Window Help
14
12
10
8
6
4
2
0
11
10
9
8
7
6
5
4
3
3.5
4
4.5
5
5.5

Figure 22.
Window points after geometric segmentation.

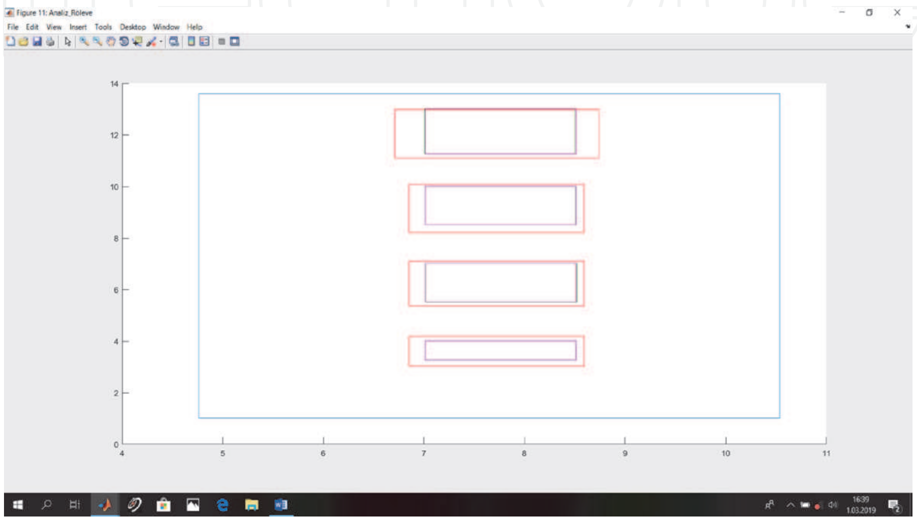

Figure 23.
Windows survey via two methods in MatLab.

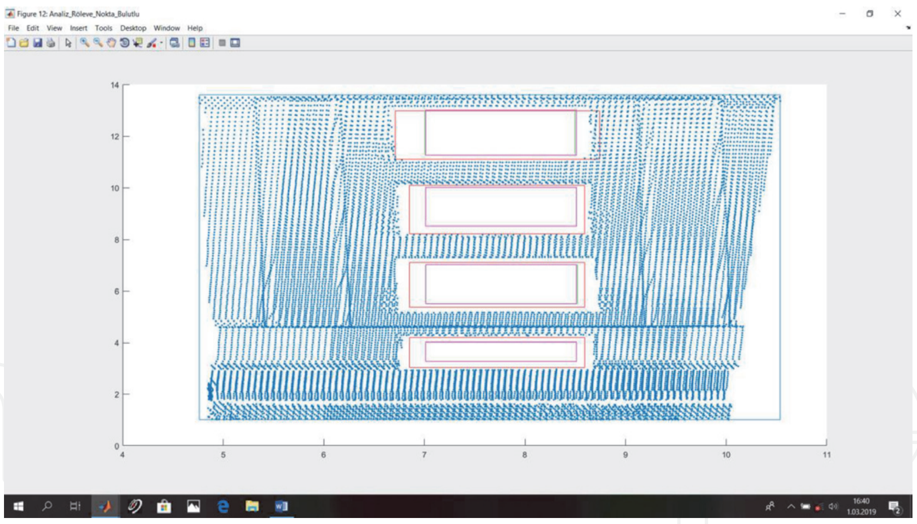
Figure 12: Analiz_Röleve_Nokta_Bulutlu
File Edit View Insert Tools Desktop Window Help
14
12
10
8
6
4
2
0
4
5
6
7
8
9
10
11

Figure 24.
Window gaps result of segmentation methods in survey point cloud in MatLab for window gaps.

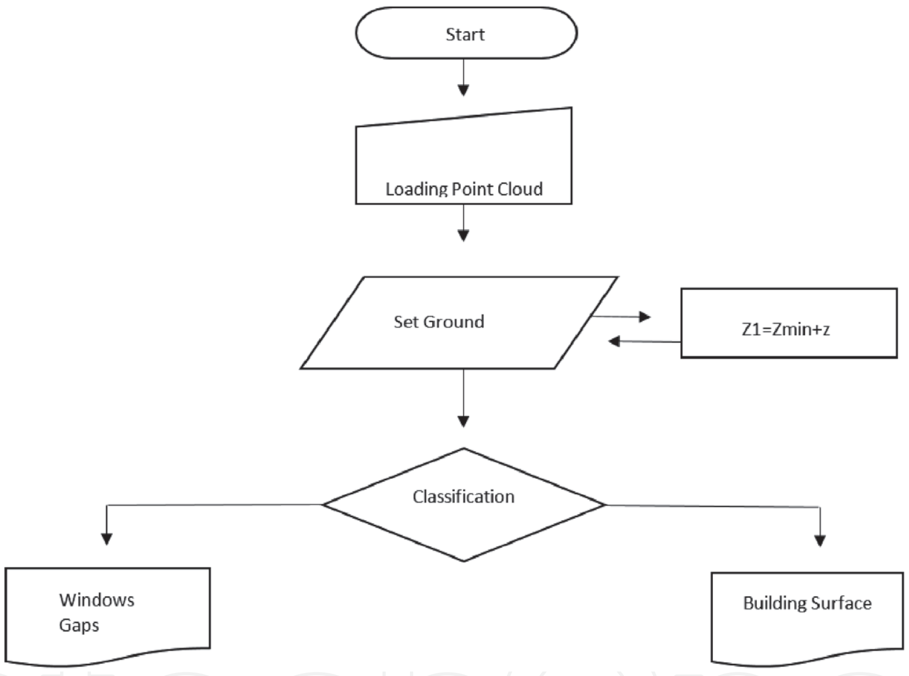

Figure 25.
Flowchart of Matlab coding for object segmentation.

had 2132, 646, 624, 383 points respectively, were created and the data points to be segmented were plotted in the **Figure 24**.

Matlab coding flowchart for this application is given in **Figure 25**.

Matlab Code of Flowchart in **Figure 25** has been given in Appendix B.

4. Conclusion

With this chapter, we observe that point cloud data need fundamental process flow as the fundamental steps of geometric segmentation by Matlab programming. Obviously, Matlab programming which depend on various algorithm by data structures. Geometric segmentation methods should be based on these alghorithm by programmers in the point cloud data. The process examples and some tips are given by Matlab coding in this chapter, which is made for some kind of point cloud segmentation flowcharts, and an approach to solve fundamental geometric segmentation is presented. Generaly, point cloud data coordinate system does not defined universal coordinate system. Thus, point cloud data listed disorderly by Cyclone 5.2 software are classified in a surface-based way, so it passes through a geometric segmentation before transformation to universal coordinate system.

In the potential, upcoming studies, radiometric and hiybrid segmentation methods might be used in Matlab coding. More accurate results might be obtained by using another method than geometric segmentation technique or by adding a third filter to geometric filtering or other segmentation methods whic are mentioned in this chapter.

Appendix A

```
clear all;
clc;
format long g;
load 'YUZEY1.txt';
load 'ham_data.txt';

I=ones(4,1);
N=transpose(YUZEY1)*(YUZEY1);
n=inv(N)*I;
X=inv(N)*(I);
[t,k]=size(ham_data);
int i;
int j;
i=1;
j=1;
matris_X=ones(t,1);
matris_Y=ones(t,1);
matris_Z=ones(t,1);

for j=1:k;
   i=1:t;
      c=ham_data(i,j);
      if j==1
         matris_X(i,1)=c;
      elseif j==2
            matris_Y(i,1)=c;
         elseif j==3
               matris_Z(i,1)=c;
            end
end
a=X(1,1);
b=X(2,1);
c=X(3,1);
d=X(4,1);
T=[ham_data];

[ns,ss]=size(T);

for i=1:ns;
```

```
   Matris_Oran(i,1) =(abs(a*(T(i,1)+ b*T(i,2)+ c*T(i,3)+ d)))/sqrt(a*a+b*b+c*c);
end

Kesme_Kriteri = std(Matris_Oran);
%——————————————————————————————————————
%——————————————————————————————————————
%——————————————————————————————————————

kes = Kesme_Kriteri;
gebze = max(Matris_Oran);
cayir = min(Matris_Oran);
yuksek = mean(Matris_Oran);
%sayi = ((Ayrac-1)/kes);
for i=1:ns;
   Matris_YOran(i,1) =(Matris_Oran(i,1)/yuksek);
end

ayar1 = max(Matris_YOran);
ayar2 = min(Matris_YOran);
ayar3 = mean(Matris_YOran);
ayar4 = std(Matris_YOran);

for i=1:ns;
   Matris_KOran(i,1) =exp(Matris_YOran(i,1));
end

dayar1 = max(Matris_KOran);
dayar2 = min(Matris_KOran);
dayar3 = mean(Matris_KOran);
dayar4 = std(Matris_KOran);

%yüzey sayisinin belirlenmesi————————————————————————

say = dayar1 - dayar2;
say = round (say);
yuzey_sayisi= say;

%——kesme kriteri ve sýnýflandýrma———————————————

kes=dayar2 + 0.25;
for i=1:ns;

    if (Matris_KOran(i,1) < kes)
                  Matris_KEGIT(i,1) = 1;
    else
                  Matris_KEGIT(i,1) = 0;
   end

end

for i=1:ns;
   if ((2.25 < Matris_KOran(i,1))&&( Matris_KOran(i,1) < 3.25))
              Matris_KEGIT(i,1) = 2;
    end
end

for i=1:ns;
    if ((3.25 <= Matris_KOran(i,1))&&( Matris_KOran(i,1) < 4.25))
                Matris_KEGIT(i,1) = 3;

    end
end

for i=1:ns;
    if ((4.25 <= Matris_KOran(i,1))&&( Matris_KOran(i,1) < 5.50))
                Matris_KEGIT(i,1) = 4;
    end
end

%————————————————————————————————————————————————————————————————
%————————————————————————————————————————————————————————————————

KY=ham_data(:,2);
KX=ham_data(:,1);
KZ=ham_data(:,3);
%Fig1=plot(KX,KY,KZ,'–rs');

T=[T,Matris_Oran,Matris_YOran,Matris_KOran,Matris_KEGIT];

Fig2 = plot(Matris_KOran,'-.or');
```

Appendix B

```
  clc;
clear;
format long g;

%Point cloud data loading

[dosyaadi,dosyayolu] = uigetfile(...
    {'*.dat;*.txt;*.xyz;*.pts','Lazer Veri Dosyaları...(*.dat,*.txt,*.xyz,*.pts)';
    '*.dat', 'Data_Dosyalar (*.dat)';...
    '*.txt', 'Txt_Dosyalar (*.txt)';...
    '*.xyz', 'Nokta_Dosyalar (*.xyz)';...
    '*.pts', 'Nokta Bulutu_Dosyalar (*.pts)';
    '*.*', 'Tüm Dosyalar (*.*)'},...
    'Bir Lazer Tarama Veri Dosyası Seçiniz:');

  if dosyaadi~=0
h=waitbar(0,'Lazer Verisi Yükleniyor');
for i=1:10

  Ham_veri_Matris = load([dosyayolu,dosyaadi]);

  [ns, ss] = size(Ham_veri_Matris);
  waitbar(i/10,h);
end

  close(h);
end

  %Segmentation starting

  X =Ham_veri_Matris(:,1);
Y =Ham_veri_Matris(:,2);
Z =Ham_veri_Matris(:,3);

  figure('Name','3D Laser Point Cloud','NumberTitle','on')
scatter3(X,Y,Z,'.');

  str1=num2str(ns);
uiwait(msgbox({'Toplam Tarama Nokta Sayısı',[str1]},'Success'));
fprintf('Toplam Tarama Nokta Sayısı %d\n',ns);

  matris_sinif=ones(ns,1);
Matris_segmentation = [X,Y,Z,matris_sinif];

  %Elevetion extraction
zemin=min(Z);

  %Elevation determined
  ifade={'Zemin Yüksekliği Değerini Giriniz !'};
baslik='Zemin Kalınlığı (birim m)';
normal={'0.35'};
zemin_kln=inputdlg(ifade,baslik,1,normal);
zemin_kln=str2double(zemin_kln);
Z1= zemin+zemin_kln;

  k=0;
for i=1:ns

      if (Ham_veri_Matris(i,3) <=Z1)
            Matris_segmentation(i,4)=0;
      k=k+1;
     end
end

for i=1:ns

    if (Matris_segmentation(i,4)== 0)

        Matris_Yer(i,1)=Ham_veri_Matris(i,1);
      Matris_Yer(i,2)=Ham_veri_Matris(i,2);
      Matris_Yer(i,3)=Ham_veri_Matris(i,3);

    end

end
```

```
  Yer_X=Matris_Yer(:,1);
Yer_Y=Matris_Yer(:,2);
Yer_Z=Matris_Yer(:,3);

  Yer_X = nonzeros(Yer_X);
Yer_Y = nonzeros(Yer_Y);
Yer_Z = nonzeros(Yer_Z);

  Matris_Yer=[Yer_X Yer_Y Yer_Z];

  [yns,yss]=size(Matris_Yer);

  figure('Name','Zemin Noktaları','NumberTitle','on')
scatter3(Yer_X,Yer_Y,Yer_Z,'.');
hold on;

  str2=num2str(yns);
msgbox({'Zemin Noktası Sayısı',[str2]},'Success');
str3=num2str(ns-yns);
msgbox({'Bina Noktası Sayısı',[str3]},'Success');

  fprintf('Zemin Noktası Sayısı %d\n',yns);
fprintf('Bina Noktası Sayısı %d\n',ns-yns);
  %Base axes determination

  Yer_y_min=min(Yer_Y);
Yer_x_min=min(Yer_X);
Yer_y_max=max(Yer_Y);
Yer_x_max=max(Yer_X);

  Eksen_y=Yer_y_min+((Yer_y_max-Yer_y_min)/2);
Eksen_x=Yer_x_min+((Yer_x_max-Yer_x_min)/2);
Line1=[Eksen_x Yer_y_min];
Line2=[Eksen_x Yer_y_max];
Point_X=[Eksen_x
      Eksen_x];

  Point_Y=[Yer_y_min
      Yer_y_max];
Point_Z=[Z1
      Z1];

  %Segmentation matrix refreshing

  Matris_seg=zeros(size(Matris_segmentation));
k=0;
t=0;
for i=1:ns
  if (Matris_segmentation(i,4)==1)

      Matris_seg(i,1)= Matris_segmentation(i,1);
     Matris_seg(i,2)= Matris_segmentation(i,2);
     Matris_seg(i,3)= Matris_segmentation(i,3);
```

```
        Matris_seg(i,4)= Matris_segmentation(i,4);
        k=k+1;
      else if (Matris_segmentation(i,4)==0)
        Matris_seg(i,1)= 0;
        Matris_seg(i,2)= 0;
        Matris_seg(i,3)= 0;
        Matris_seg(i,4)= 0;
        t=t+1;
      end

    end
end

  %Building Points determinated

  Bina_X=nonzeros (Matris_seg(:,1));
Bina_Y=nonzeros (Matris_seg(:,2));
Bina_Z=nonzeros (Matris_seg(:,3));

  Matris_Bina=[Bina_X,Bina_Y,Bina_Z];

  [bns,bss]=size(Matris_Bina);

  sinif=ones(bns,1);

  Matris_Bina_seg=[Matris_Bina,sinif];
  %Building Points drawing

  figure('Name','Bina Noktaları','NumberTitle','on')

  scatter3(Bina_X,Bina_Y,Bina_Z,'.');hold on
plot3(Point_X,Point_Y,Point_Z, '-.o');
hold on
grid on

  %Building Points segmented

  sol=0;
sag=0;
for i=1:bns

        if (Matris_Bina(i,1) < Eksen_x )
              Matris_Bina_seg(i,4)= 1;
              sol=sol+1;
      else if (Matris_Bina(i,1) > Eksen_x )
              Matris_Bina_seg(i,4)= 2;
              sag=sag+1;
            end
    end

end

  Matris_Bina_sol=zeros(bns,3);
```

```
Matris_Bina_sag=zeros(bns,3);

 for i=1:bns
        if (Matris_Bina_seg(i,4)==2)
        Matris_Bina_sag(i,1)= Matris_Bina_seg(i,1);
        Matris_Bina_sag(i,2)= Matris_Bina_seg(i,2);
        Matris_Bina_sag(i,3)= Matris_Bina_seg(i,3);
        else
        Matris_Bina_sag(i,1)= 0;
        Matris_Bina_sag(i,2)= 0;
        Matris_Bina_sag(i,3)= 0;
        end
end

 Matris_Bina_Sag_X=Matris_Bina_sag(:,1);
Matris_Bina_Sag_Y=Matris_Bina_sag(:,2);
Matris_Bina_Sag_Z=Matris_Bina_sag(:,3);

 Matris_Bina_Sag_X=nonzeros(Matris_Bina_Sag_X);
Matris_Bina_Sag_Y=nonzeros(Matris_Bina_Sag_Y);
Matris_Bina_Sag_Z=nonzeros(Matris_Bina_Sag_Z);
Matris_Bina_Sag=[Matris_Bina_Sag_X,Matris_Bina_Sag_Y,Matris_Bina_Sag_Z];

for i=1:bns
       if (Matris_Bina_seg(i,4)==1)
       Matris_Bina_sol(i,1)= Matris_Bina_seg(i,1);
       Matris_Bina_sol(i,2)= Matris_Bina_seg(i,2);
       Matris_Bina_sol(i,3)= Matris_Bina_seg(i,3);
       else
       Matris_Bina_sol(i,1)= 0;
       Matris_Bina_sol(i,2)= 0;
       Matris_Bina_sol(i,3)= 0;
       end
end

 Matris_Bina_Sol_X=Matris_Bina_sol(:,1);
Matris_Bina_Sol_Y=Matris_Bina_sol(:,2);
Matris_Bina_Sol_Z=Matris_Bina_sol(:,3);

 Matris_Bina_Sol_X=nonzeros(Matris_Bina_Sol_X);
Matris_Bina_Sol_Y=nonzeros(Matris_Bina_Sol_Y);
Matris_Bina_Sol_Z=nonzeros(Matris_Bina_Sol_Z);

 Matris_Bina_Sol=[Matris_Bina_Sol_X,Matris_Bina_Sol_Y,Matris_Bina_Sol_Z];

 %Left side – right side deciding

 [bnsol,bnssol] = size(Matris_Bina_Sol);
[bnsag,bnssag] = size(Matris_Bina_Sag);

 if bnsol > bnsag
  msgbox('Bina, yol eksenine göre Sol taraftadır !','Success');
  [bina,sut_bina]=size(Matris_Bina_Sol);
```

```
    Matris_sinif=ones(bina,1);
    Matris_Bina_Segment=[Matris_Bina_Sol,Matris_sinif];

  %Segmented Building Points drawing
figure('Name','Bina Noktaları','NumberTitle','on')
scatter3(Matris_Bina_Sol_X,Matris_Bina_Sol_Y,Matris_Bina_Sol_Z,'.');
hold on;

else
    msgbox('Bina, yol eksenine göre Sağ taraftadır !','Success');
    [bina,sut_bina]=size(Matris_Bina_Sag);
    Matris_sinif=2*ones(bina,1);
    Matris_Bina_Segment=[Matris_Bina_Sag,Matris_sinif];

end

  %Referance surface determination

  Bina_yuzey_X_max = max(Matris_Bina_Segment(:,1));
Bina_yuzey_X_min = min(Matris_Bina_Segment(:,1));

  Bina_yuzey_Z_max = max(Matris_Bina_Segment(:,3));
Bina_yuzey_Z_min = min(Matris_Bina_Segment(:,3));

  Bina_yuzey_Y_max = max(Matris_Bina_Segment(:,2));
Bina_yuzey_Y_min = min(Matris_Bina_Segment(:,2));

  BY1=[Bina_yuzey_Z_max,Bina_yuzey_Y_max,Bina_yuzey_X_max,1];
BY2=[Bina_yuzey_Z_max,Bina_yuzey_Y_min,Bina_yuzey_X_max,1];
BY3=[Bina_yuzey_Z_min,Bina_yuzey_Y_max,Bina_yuzey_X_max,1];
BY4=[Bina_yuzey_Z_min,Bina_yuzey_Y_min,Bina_yuzey_X_max,1];

  BG1=[Bina_yuzey_Z_max,Bina_yuzey_Y_max,Bina_yuzey_X_max];
BG2=[Bina_yuzey_Z_max,Bina_yuzey_Y_min,Bina_yuzey_X_max];
BG3=[Bina_yuzey_Z_min,Bina_yuzey_Y_max,Bina_yuzey_X_max];
BG4=[Bina_yuzey_Z_min,Bina_yuzey_Y_min,Bina_yuzey_X_max];

  BG=[BG1
    BG2
    BG3
    BG4];

  BGGX=BG(:,1);
BGGY=BG(:,2);
BGGZ=BG(:,3);

BY=[BY1
    BY2
    BY3
    BY4];

%Reference surface Points
```

```
GZ=[Bina_yuzey_Z_max
   Bina_yuzey_Z_max
   Bina_yuzey_Z_min
   Bina_yuzey_Z_min];

GY=[Bina_yuzey_Y_max
   Bina_yuzey_Y_min
   Bina_yuzey_Y_max
   Bina_yuzey_Y_min];

GX=[Bina_yuzey_X_max
   Bina_yuzey_X_max
   Bina_yuzey_X_max
   Bina_yuzey_X_max];

%Reference surface drawing

  figure('Name','Bina En Büyük Referans Yüzey Noktaları','NumberTitle','on')
scatter3(GX,GY,GZ,'.');
hold on;

  figure('Name','Bina En Büyük Referans Yüzeyi','NumberTitle','on')
plot3(GX,GY,GZ);
grid on;

GZ=[Bina_yuzey_Z_min
   Bina_yuzey_Z_max
   Bina_yuzey_Z_max];

KZ=[Bina_yuzey_Z_max
   Bina_yuzey_Z_min
   Bina_yuzey_Z_min];

GY=[Bina_yuzey_Y_min
   Bina_yuzey_Y_max
   Bina_yuzey_Y_min];

KY=[Bina_yuzey_Y_max
   Bina_yuzey_Y_min
   Bina_yuzey_Y_max];

GX=[Bina_yuzey_X_max
  Bina_yuzey_X_max
   Bina_yuzey_X_max];

  figure('Name','Referans Yüzey','NumberTitle','on')
fill3(GX,GY,GZ,GX);
grid on
hold on
fill3(GX,KY,KZ,GX)
grid on
hold on
```

```
  dlmwrite ('Bina_Referans_Yuzey_noktaları.txt',BY,'delimiter','\t');
I=ones(4,1);
N=transpose(BY)*BY;
n=pinv(N)*I;
X=pinv(N)*(I);

  %Reference surface parameters

  a=X(1,1);
b=X(2,1);
c=X(3,1);
d=X(4,1);

PSag=[a
    b
    c
    d];

  dlmwrite('Bina_Referans_Düzlem_Par.txt',PSag,'delimiter','\t');
%Building surface segmentation

for i=1:bina
    Derinlik_Bina(i,1)=(-(d+b*Matris_Bina_Segment(i,2)+a*Matris_Bi-
    na_Segment(i,3))/c);
    Delta_Derinlik(i,1)= Derinlik_Bina(i,1)- Bina_yuzey_X_max;
    DDEXP_Bina(i,1) = 1/(exp(Delta_Derinlik(i,1)));
    Egim_Acisi_Bina(i,1) = atan((Matris_Bina_Segment(i,2)-Z1)/(Matris_
Bina_Segment(i,1)- Eksen_y));
end

  Yuzey_Der=(Bina_yuzey_X_max+Bina_yuzey_X_min)/2;
Filtre_Yuzey = Yuzey_Der*ones(bina,1);
Matris_Bina_Segment=[Matris_Bina_Segment,Derinlik_Bina,Delta_Derinlik,
DDEXP_Bina,Filtre_Yuzey];

  KOD=[Matris_Bina_Segment(:,2) Matris_Bina_Segment(:,3) Matris_
Bina_Segment(:,8)];
dlmwrite('Geo_Filitre_Giriş.txt',KOD,'delimiter','\t');

  %Geometric Segmentation start
run Geometric.m;
%Geometric Segmentation finish

KSM = dlmread('bosluk_output.txt');

hold on;

  figure('Name','Geometric Öncesi Yüzey Noktaları','NumberTitle','on')
scatter3(Matris_Bina_Segment(:,8),Matris_Bina_Segment(:,2),Matris_
Bina_Segment(:,3),'.');hold on
scatter3(KSM(:,3),KSM(:,1),KSM(:,2),'.','r');
hold on;
```

```
 Ref_Nok_X=Point_X/2;
Ref_Nok_Y=Point_Y/2;
Ref_Nok_Z=Z1;

 %Building flat number info ask

 ifade1={'Bina Kaç Katlı ?'};
normal1={'4'};
baslik1='Bina Kat Sayısı';
satir1=1;
cevap1=inputdlg(ifade1,baslik1,satir1,normal1);
Bina_Kat_Sayisi=str2double(char(cevap1(1,1)));

 %Flat Classing

 [kfns,kfss]=size(KSM);
Egim_Acisi_Geo=ones(kfns,1);
EXP_Kal=ones(kfns,1);
for i=1:kfns
  Egim_Acisi_Geo(i,1) = atan(KSM(i,2)/10);
  EXP_Kal(i,1)=exp(Bina_Kat_Sayisi*Egim_Acisi_Geo(i,1));
end

 std_Kal=std(EXP_Kal);
max_Kal=max(EXP_Kal);
min_Kal=min(EXP_Kal);
KAL_Seg_Deg = round(std_Kal,0)/(Bina_Kat_Sayisi*0.5);
Seg_Kal=[KSM Egim_Acisi_Geo EXP_Kal];

 for i=1:Bina_Kat_Sayisi
  eval(sprintf('Pen%d = NaN(kfns,3)', i));
end

for j=Bina_Kat_Sayisi:-1:1
  for i=1:kfns
    if Seg_Kal(i,5)>KAL_Seg_Deg*2^(j-2) && Seg_Kal(i,5)<KAL_Seg_Deg*2^
    (j-1)
    eval(sprintf('Pen%d(i,1)=Seg_Kal(i,1)', j));
    eval(sprintf('Pen%d(i,2)=Seg_Kal(i,2)', j));
    eval(sprintf('Pen%d(i,3)=Seg_Kal(i,3)', j));
   end

  end
PP=snip(eval(sprintf('Pen%d', j)),nan);
eval(sprintf('PP%d = PP', j));

 PX=[max(eval(sprintf('PP%d(:,1)', j)))
  min(eval(sprintf('PP%d(:,1)', j)))
  min(eval(sprintf('PP%d(:,1)', j)))
  max(eval(sprintf('PP%d(:,1)', j)))
  max(eval(sprintf('PP%d(:,1)', j)))];

   eval(sprintf('aax%d=PX(1,1)',j));
```

```
     eval(sprintf('bbx%d=PX(2,1)',j));

     eval(sprintf('PP%dX = PX', j));

   PY=[min(eval(sprintf('PP%d(:,2)', j)))
     min(eval(sprintf('PP%d(:,2)', j)))
     max(eval(sprintf('PP%d(:,2)', j)))
     max(eval(sprintf('PP%d(:,2)', j)))
     min(eval(sprintf('PP%d(:,2)', j)))];

      eval(sprintf('aay%d = PY(3,1)',j));
     eval(sprintf('bby%d = PY(1,1)',j));

      eval(sprintf('PP%dY = PY', j));

   PZ=[eval(sprintf('PP%d(1,3)', j))
     eval(sprintf('PP%d(1,3)', j))
     eval(sprintf('PP%d(1,3)', j))
     eval(sprintf('PP%d(1,3)', j))
     eval(sprintf('PP%d(1,3)', j))];
     eval(sprintf('PP%dZ = PZ', j));

 end

   KODX=[max(KOD(:,1))
     min(KOD(:,1))
     min(KOD(:,1))
     max(KOD(:,1))
     max(KOD(:,1))];

   KODY=[min(KOD(:,2))
     min(KOD(:,2))
     max(KOD(:,2))
     max(KOD(:,2))
     min(KOD(:,2))];

   KODZ=[KOD(1,3)
     KOD(1,3)
     KOD(1,3)
     KOD(1,3)
     KOD(1,3)];

   %Results drawing

   figure('Name','Röleve','NumberTitle','on')
 for i=1:Bina_Kat_Sayisi
 line(eval(sprintf('PP%dX', i)),eval(sprintf('PP%dY', i)),eval(sprintf('PP%dZ',
 i)),'Color','red');hold on
 end
 line(KODX,KODY,KODZ);hold on
```

Author details

Bahadır Ergün* and Cumhur Şahin
Department of Geomaics Engineering, Gebze Technical University, PK 141, Gebze 41400 Kocaeli, Türkiye

*Address all correspondence to: bergun@gtu.edu.tr

References

[1] Ergun B. A novel 3D geometric object filtering function for application in indoor area with terrestrial laser scanning data. Optics &Laser Technology. 2010;**42**:799–804.

[2] Sahin C. Planar segmentation of indoor terrestrial laser scanning point clouds via distance function from a point to a plane. Optics and Lasers in Engineering. 2015;**64:**23–31. DOI: 10.1016/j.optlaseng.2014.07.007.

[3] B. Ergun, C. Sahin, A. Aydin. Two-dimensional (2-D) Kalman Segmentation Approach in Mobile Laser Scanning (MLS) Data for Panoramic Image Registration, Lasers in Engineering, 2020; **48:** 121–150.