AUTOMATIC GENERATION OF FLOOR PLANS FOR INDOOR NAVIGATION USING POINT CLOUD

Ullas Rajvanshi [1], Sandra Verhagen[2], Robert Voûte[3,4]

[1] Internship Trainee, Geoscience and Remote Sensing, Civil Engineering and Geosciences Faculty, TU Delft, The Netherlands
U.Rajvanshi@student.tudelft.nl
[2]University Supervisor, Geoscience and Remote Sensing, Civil Engineering and Geosciences Faculty, TU Delft, The Netherlands
A.A.Verhagen@tudelft.nl
[3] Company Supervisor, Vice President Consulting Geo-ICT, CGI Nederland BV, Rotterdam, The Netherlands
Robert.Voute@cgi.com
[4]Department of GIS Technology, Faculty of Architecture and the Built Environment, TU Delft, the Netherlands
R.Voute@tudelft.nl

KEY WORDS: Indoor Positioning, Point cloud, Laser scanning, Indoor reconstruction, Automatic floorplan generation, Analysis

ABSTRACT:

This paper describes a methodology for an automated floor plan generation using point cloud captured via Mobile Laser Scenner(MLS) for indoor navigation purposes. The method takes as input a 3D point cloud and using geometrical features, a ground plane histogram is computed representing the points density using projection. Finally using edge extraction and Hough Transform the polylines are extracted out in a very robust and efficient manner. This methodology works for any cluttered environment due to the projection and obtaining a volumetric slice where the extra noise is then removed using cluttering techniques. The methodology tested on a hallway of acadmic building gives promising results.

1. INTRODUCTION

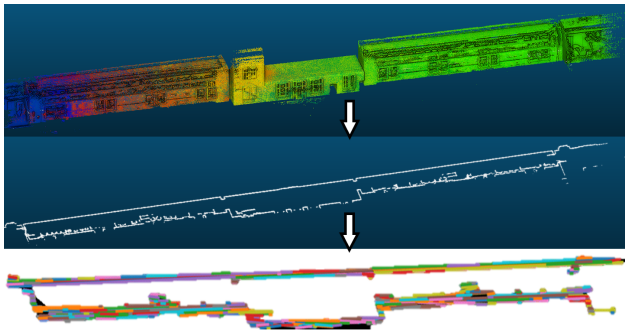

Figure 1. Automatic floorplan modelling: Input Point Cloud (*Top*) captured using MLS, the method seeks and detect possible walls based on density, volumetric slicing, projection (*Middle*), to find resulting polyline in the floor plan (*Bottom*)

Floor plans are a vital part of any architectural piece and helps to visualise the whole building and becomes of utmost importance when considering the scope of Indoor Positioning System (IPS) with indoor applications such as ArcGIS Indoors. However, most of the buildings have outdated floor-plans which can provide wrong directions during the navigation in indoor spaces or are changing due to undocumented renovations. Laser scanners have now become the new tool to make the 3D model of the buildings, however still utilises a lot of manual steps in order to achieve the final floor-plan (in 2D polygon shapefile) which can be then imported into ArcGIS Indoors and can be used for indoor navigations.
The goal of this paper is to automatically generate the floor plans for seamless navigation in indoor spaces where the input is point cloud scanned using a Mobile Laser Scanner. The following research was carried out during the two months internship at CGI Nederland in Rotterdam, The Netherlands.
There are mutiple algorithms like the one proposed by (Okorn et al., 2010) which also bases the following floor plan generation using the same principle, but some which use segmentation techniques for fast 3D line segment detection instead of using 2D plane (Cui et al., 2019). Furthermore with the advancement in machine learning, neural nets are being trained in order to detect the floorplans (Liu et al., 2018). However they all require either a large amount of training data or use heavy computation power to process the results. The current method makes use of geometric attributes in the point cloud data and using computer vision techniques to provide the results.
The approach for the following principle is quite simple yet efficient. Since the idea for creating these floorplans is based on walls, the floors are removed and object dicing is done to obtain a 3D volumetric slice which is then projected onto 2D plane to compute the histogram density. Finally linear structures based on the given parameter values are detected using Hough Transform approach. It is to be noted that the extension of this method towards detecting non-plan and non-vertical structures if for future work (Okorn et al., 2010).

2. DATA AND TOOLS

The current dataset used in order to test the methodology proposed in the paper is captured by CGI Nederland and used under their permission. The current methodology is designed to be working for all the indoor environments with planar and vertical strcutures such that they were built on a cartesian grid. The dataset used throughout this paper is point cloud of the hall of The Faculty of Architecture and the Built Environment is considered. The dataset is visualised using the PPTK Viewer Library in Python (B.V., 2018) as can be seen in Fig. 2. The reason

the data is being visualised in this library is due to robustness and ability to load 100M points with 20fps is far better than visualising the same in software such as CloudCompare. The data used in the following research is captured with a hand-held Mobile Laser Scanner (MLS), which also saves the path (trajectory) taken in the building (Flikweert et al., 2019).As can be seen in the isometric view of the point cloud in Fig. 2 there is a lot of clutter and one of the main challenges to automate the generation of the floor plans is handling of the clutter. Clutter is defined as any 3D data which is not relevant for the final output such as furniture, lights and interior decorations (Okorn et al., 2010), and while scanning an indoor environment the clutter is quite large and spread around. However it will still have less point density than the walls, ceiling and floors which are of utmost importance when modelling the floor plan.

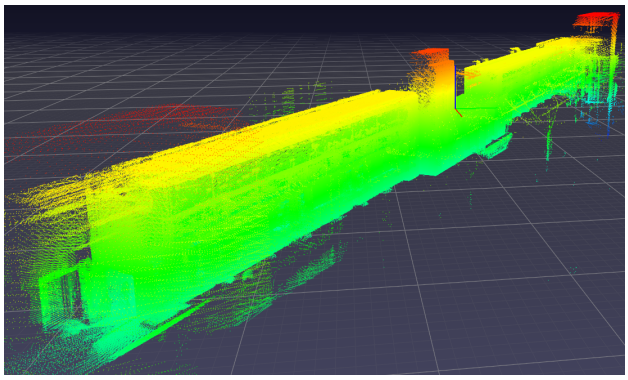

Figure 2. Front isometric view of the raw unprocessed point cloud

3. METHOD

The currently methodology proposed in this research is divided into steps as shown in Fig. 3. The algorithms have either been created during the research or made use of open source available algorithms. The input to the method is a raw point cloud in LAS file type and the produced out is the floor plan with the polylines. The methods are as follows: In order to start the floor plan extraction, the first approach is to get rid of the noise which is the points not included inside the building. This is done by first dividing the point cloud into small parts with a grid step of every 100 points and then separation of the ground from the rest (Lo, Chenb, 2012). The excess noise is then removed by using noise filter statistical outlier removal technique. The floor removal is then processed using the histogram distribution of the points and to leave out walls from the dataset with extracting ceiling and floor within horizontal structures and eliminating the clutter from the remaining cloud by use of estimating surface normal and filtering points whose normal are not parallel in the vertical direction (Okorn et al., 2010) (Babacan et al., 2016).

The next step is to operate a volumetric slicing from the processed point cloud. In order to do this, a threshold needs to be determined which can be optimised and adapted from the histogram distribution after the filtering (Previtali et al., 2014) and the floor removal. This is done based on the height where no clutter is to be found usually at the height between 1.95 and 2.05 meters. Following the volumetric slicing, Density-based spatial clustering of applications with noise (DBSCAN) is performed which is a data clustering algorithm and groups together

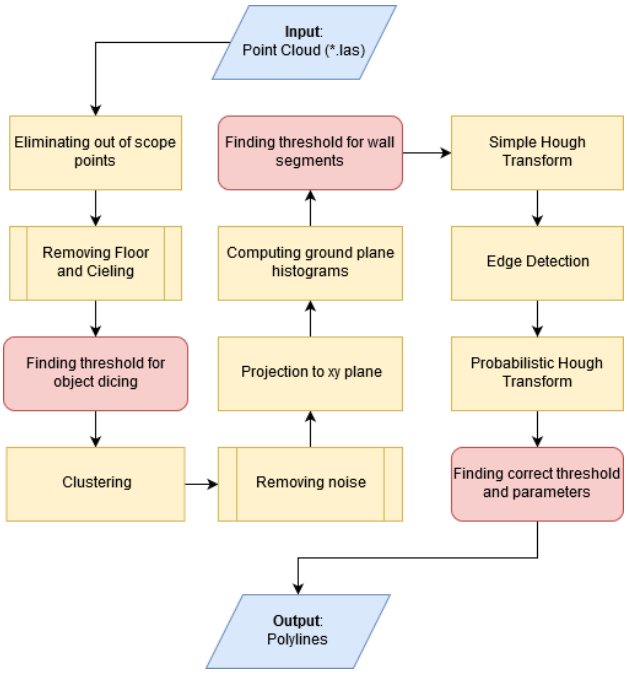

Figure 3. Process flow

points that are within a specified region. The clutter with least amount of points are then removed in order to get rid of noise. The obtained point cloud (in 3D) is then projected onto an xy plane to get a 2D image out of the point cloud. This approach is carried out since when these 3D points are projected onto the xy plane, the point density of the projected points is expected to be highest at the wall locations since walls are the most scanned segments in the point cloud. The projected points are then used to create 2D histograms of points projected into ground plane (Okorn et al., 2010).

Given the following ground plane histogram, the next project is to mask the area using the pixel count. This is done by applying a threshold to the projected pixel count, where the wall structures can then be represented with the highest spectrum of the colorbar and will be seen with much more ease. Then the approach of Simple Hough Transform is first applied to the obtained image . Hough Transform is useful for detecting any parametric curves like lines and conics and is relatively unaffected by the gaps in the curves and noise for the point cloud (Hough, 1962)(Okorn et al., 2010). The algorithm for line detection by Hough transform is as follows:

Consider a line which has two parameters (m, b), where m is the slope of the line and b is the y-intercept as shown in Fig. 4. For a given point (x_0, y_0), the lines that could pass through the point are all (m, b) satisfying

$$y_0 = mx_0 + b \tag{1}$$

Then using the quantise parameter space (m, b), the accumular array $A(m, b)$ is created and then set equal to zero for all (m, b). For each image edge (x_i, y_i) increment :

$$A(m, b) = A(m, b) + 1 \tag{2}$$

If (m, b) lies on the line then:

$$b = -x_i m + y_i \tag{3}$$

Finally the local maxima in $A(m, b)$ is found which indicates

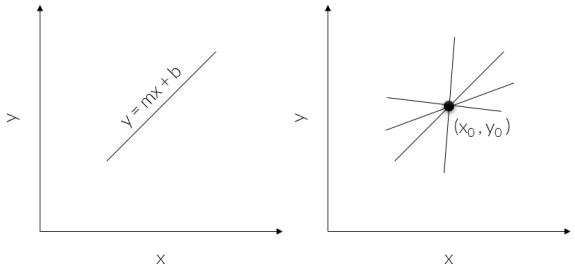

Figure 4. A line with equation $y = mx + b$ (left), lines passing through a point (x_0, y_0)

the parameters of the most prominent lines in the input image. Peaks can be found most easily by applying a threshold or a relative threshold (values equal to or greater than some fixed percentage of the global maximum value) (Kiryati et al., 1991). However there is a need of optimisation since the slope m is undefined when the line is vertical and hence hough space (θ, ρ) is used in order to solve this issue. This is very well explained in (Kiryati et al., 1991) by the author. The pseudo code for the following approach is as follows:

Algorithm 1 Hough Transform

```
for all x do
    for all y do
        if (x_i, y_i) in (x, y) then
            for all θ do ρ = x cos(θ) + y sin(theta)
                increment cell in H corresponding to (θ, ρ)
```

4. EXPERIMENTAL RESULTS

The data used to evaluate the algorithm as mentioned in Section 2 is the hall library of the architecture faculty in Delft, The Netherlands. The building was selected because of its Manhattan layout. The processed results from the following building can be found in detail in the next several sub-sections.

4.1 Elimination of out-of-scope points

In Fig. 5, on the left the whole dataset is represented in a one dimensional (z) histogram distribution plot where the large maxima at the bottom most part (≈ 0 meters) shows the floor and another peak at around 3.1 meters indicates the ceiling. The peaks in the middle of this threshold range consists of clutter at each elevation. After the floor removal, it can be seen that from the figure in the right of Fig. 5 that the whole floor has been removed, along with the excess noise at each elevation step. The result of this process can also be seen in Fig. 6 where on the left is the original dataset and on the right is the processed dataset with removed floor. This dataset is being called as the *filtered dataset* throughout the report. Even though fom the histogram looks like there is no value lower than 1.5 meters however, due to the limit of the y-axis the data cannot be seen. However, the frequency of the points lower than 1.5 meters is indeed much lower than that of the original dataset. One reason for this is planar structures like tables which also represents similar structure like floor might have been disregarded. Upon the filtering operation the original dataset which consisted of $6,887,167$ points was reduced to about $3,403,718$ points. Unlike a terrestrial laser scanner where multiple scans overlap result into

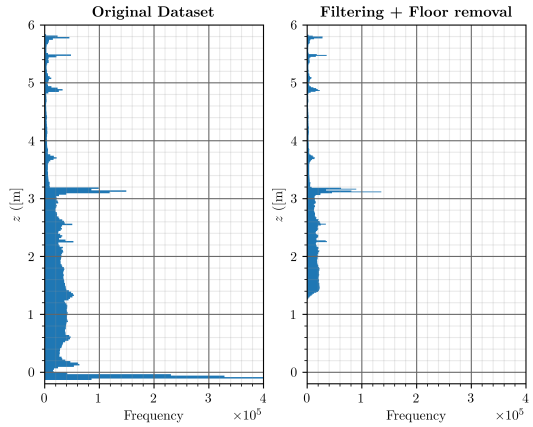

Figure 5. The height histogram against the height of the dataset

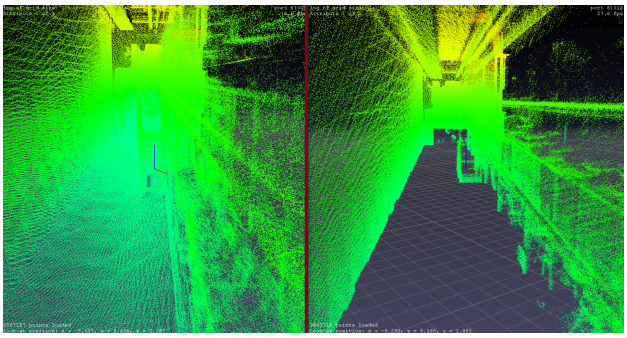

Figure 6. The raw dataset *on the left* and the processed dataset *on the right* with the floor removal

densely sample regions, this method. The filtered data is also used since the floor and the top most part of the buildings (the peak in the height histogram) are not used for the part of wall detection. The floor and the ceiling heights obtained from the histogram are then also important for selecting the threshold for the cross section. From this filtered dataset and the threshold obtained, the results of the volumetric slicing can be seen in Fig. 7. The general outline of the floorplan of the building can already be seen from this obtained point cloud. Throughout the whole report, this datatset will be called as *sliced dataset*. This dataset is overlaid on the original dataset to show the final sliced point cloud data in Fig. 8.

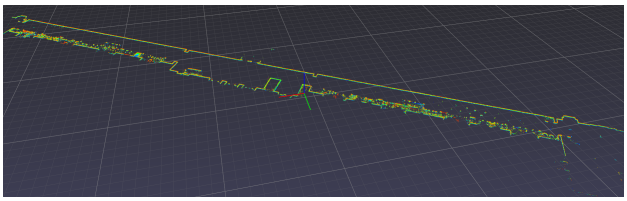

Figure 7. Volumetric Slicing of the filtered data based on the histogram and floor removal

4.2 Ground Plane Histogram

The original, filtered and the sliced data is then projected onto the xy plane just like the histogram distribution, however this time using the two dimensional image and the count of each

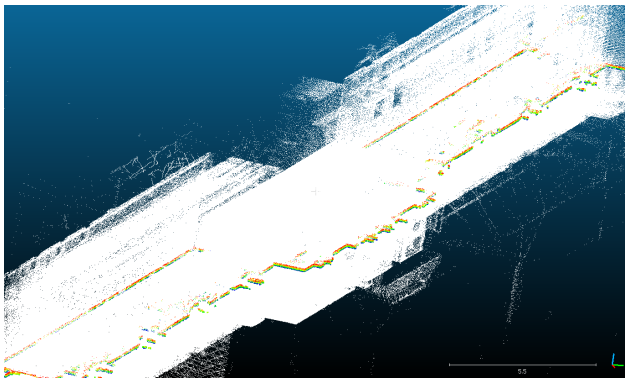

Figure 8. Sliced data (shown in color) from the original dataset

point along the z dimension. The 2D histogram can be seen in Fig. 9 for each dataset. The reason to show the results of all the three datasets is to identify the difference of the output image based on the processing steps and the necessity of these steps. As can be seen from the first image of the original dataset the histogram distribution is quite spread since in this case walls are not the only part of the scanned building with the most amount of points. The floor and the ceiling can be very well observed by the bright spots in the middle of the figure. In the second figure, the filtered data is projected and already a significant decrease in the count can be seen as there is now no floor and ceiling interfering and the walls can already be seen. Finally the sliced dataset can be observed in the third image where it is a bit difficult to observe due to the fact that projection was carried out for only 10 cm thick volumetric slice and the points count has decreased a lot.

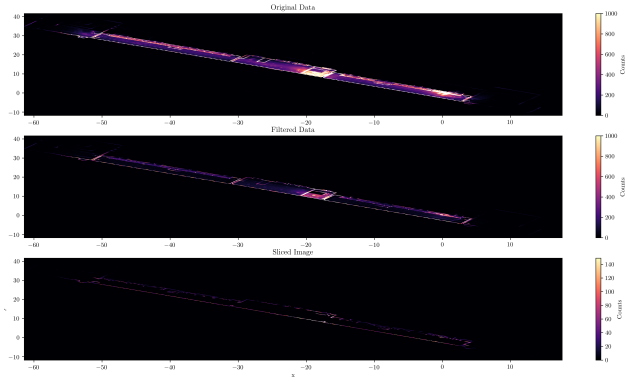

Figure 9. Ground plane histograms of the projected counts of point into xy plane

4.3 Walls Detection

The obtained 2D histograms are then converted into 2D image and also masked in order to remove low density cells based on the threshold value obtained from the histogram. This results into the walls observed in the image as shown in Fig. 10. The threshold values obtained for the original and the filtered dataset were 900 and 200 counts respectively and all values lower than that were disregarded. The sliced dataset was masked with values lower than 8, however since the total count is much lower than the other dataset, this threshold does play an important role in order to get rid of excess clutter. The threshold obtained for the filtered dataset was not that good and hence was manually updated in order to get a better wall segment. It is important to mention that the following result is still in 2D image from the projected point cloud.

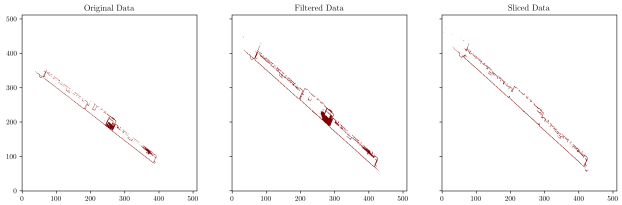

Figure 10. Wall segment detection by masking the ground plane histogram to remove low density cells

4.4 Hough Transform

To evaluate and predict the (poly-)lines from the 2D image, hough transform is applied. The results from simple Hough transform were just two straight lines on the border of the point cloud and hence are not shown here, however the probabilistic hough transform was then applied and the results for the original dataset with different parameters and the sliced dataset can be seen in figures 11, 12 and 13. The left image shows the input image which is the obtained array from the wall segment detection. The middle image is the result of the edge detection algorithm which results a binary image where everything in black is 0 meaning no edges and everything in white is 1 indicating edges.

This edge extraction is done use **Canny Edge Extraction** algorithm where multiple parameters such as the sigma where lower values will result in higher noise, and applying a threshold boundary to the following input image in order filter out edge pixels with a weak gradient value and preserve edge pixels with a high gradient value. (Canny, 2019). The output of the edge detection is an array of edge pixels $[(x_1, y_1), (x_2, y_2) \dots (x_n, y_n)]$. The parameters used for each of the images is available in the description of each result. The right image is the final result of the polylines detected from the edges image. Each line is given a random colour and the black background is only for visualisation purposes to highlight the lines.

Just like the edge extraction, this also has several parameters which could be changed based on the obtained input image. The parameters used to generate the following images are threshold on the density histogram image, the length of the line hence representative of the real world scenario and the line gap which is the minimum distance between points that can be included in the same detected line. The parameters were chosen based on the experiments and the inspection of the point cloud. The resulting polylines are stored as an array with $[(x_1, y_1), (x_2, y_2)]$ coordinates of the line segments.

In Fig. 11, the input image is the filtered dataset, and the canny edge extraction parameters chosen for the current dataset are $\sigma = 3$ since the dataset still has some noise which needs to be filtered out and threshold between 10 and 20 were chosen, After extracting the binary value edges, hough transform was applied with the following parameters, threshold $= 10$ with line length 15 and line gap 5.As can be seen the structure of the building is already now visual from the polylines. This is quite remarkable since the processed or even the original dataset can be used in order to provide the outline structure of the building.

In Fig. 12, the input image is the filtered dataset, however the paramters for the edge extraction and the hough transofrm have changed, $\sigma = 2.5$ in order to include some more structure since upon using $\sigma = 3$ some parts of the building were lost. The

low threshold is still the same, however the high threshold has been increased to 30. All the hough transform parameters are still kept the same except for the line gap which was increased to 20. The result of that can already be seen on the bottom part of the image where the two lines have been merged together.
In Fig. 13, the processed sliced data was considered as the input image. In order to extarct the edges, the parameters chosen for the canny edge extraction algorithm were $\sigma = 1$ and threshold values between 10 and 15. Finally for the hough transform to estimate the lines the threshold was chosen to be 10 with line length of 10 and line gap of 10. It can be seen that this provides the best polylines approximation for the following dataset since the structure and the layout of the point cloud is preserved when operating with the following parameter values

4.5 Performance on the dataset

When dealing with point clouds, it if utmost importance to mention about the time it takes from the starting of the algorithm to the end. This is in order to understand how the runtime will change depending on whether there are more points or less points. The whole algorithm is programmed in Python and with no parallel programming and the total time taken by the program to obtain the end result is 7 minutes and 46 seconds on i7-7820QH processor with 16GB RAM. This is remarkable processing speed to obtain polyline for dataset which is more than 6 million points. Almost half of the processing time is spent on obtaining the filtered dataset since it is evaluating a floorplan at evey 1 meter with grid step of 100 points.

4.6 Ground Truth

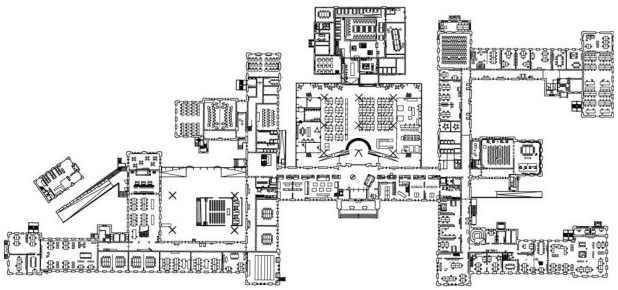

Figure 14. Ground truth map of Architecture faculty (Roos, Braaksma, 2011)

In order to validate the relative accuracy of the lines obtained with the actual floorplan, a floorplan created in atleast 2011 as shown in Fig. 14 is being used. However upon visual inspection of the building the floorplan is not to the most accurate but the outline of the structure still remains the same. The point cloud scanned is from the lower left part of the building as shown in Fig. 15. Upon comparing the ground truth floorplan to the polylines floorplan obtained from the method it can be seen that the central long hallway has indeed detected remarkably smooth despite the clutter inside the original scan. Furthemore the walls detected has given the floorplan another level of detail when considering this case for first responders. The square shaped area on the left and the right of the floorplan, wasn't detected well since the point cloud obtained at this data was missing. However interpolation with line gap as shown in Fig. 12 is almost able to capture this dataset even though the data is still missing.

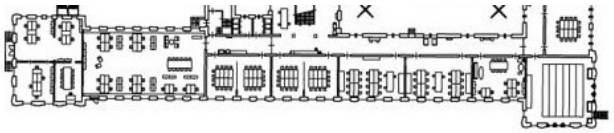

Figure 15. Lower left part of the ground truth map of Architecture faculty for which the point cloud is captured (Roos, Braaksma, 2011)

5. CONCLUSIONS

This study proposed a method in order to get the structure lines in form of polylines out of an unorganised point cloud. These structured lines can then be used in order to create floorplans for applications like ArcGIS Indoors and navigate seemlessly in an indoor environment. Over the last year, indoor navigation has been attracting interest over multiple domains but obtaining a fast and reliable model has been very difficult. This paper presented a highly robust method for automatic floor plan generation using the wall structure despite the clutter present from point cloud using the geometric properties. The computation of ground plane histogram and removal of the floor significantly effected the result since most of the noise was cluttered into one and removed resulting into a far cleanear point cloud which could be further processed. Furthermore, in order to evaluate the quality of the results the obtained floor map was compared with the ground truth.

6. RECOMMENDATION AND FUTURE WORK

The current methodology depends on certain parameters which are required to be input manually based on visual inspection. However the threshold could be adapted for this and predicted to provide the best parametric value. Moreover the current methodology uses 2D projection of the 3D dataset, however come algorithms prepared in C++ are able to detect 3D line segments very fast (Cui et al., 2019) with very simple yet efficient segment detection.
In the future, the expressed method can be further improved by firstly joining or merging these polylines into one outline which can be then expressed from point cloud coordinates to the geodetic coordinates so that it could be put on a map. There are a lot of algorithms for Line Simplification with the Douglas-Peucker Algorithm (Saalfeld, 1999) which can also be applied to get one simplified line out of the multiple poly-lines. The next step would be to merge the findings of the work done by (Flikweert et al., 2019) and attach the doors in the final floorplan and extending the methodology for detection of windows

ACKNOWLEDGEMENTS

The author would sincerely like to express his gratitude toward Dr. Linh Truong Hong for his support and help throughout the whole internship project. Also sincere thanks to Robert Voûte from CGI Nederland for giving me the opportunity to work and Dr. Sandra Verhagen for providing her supervision towards the project.

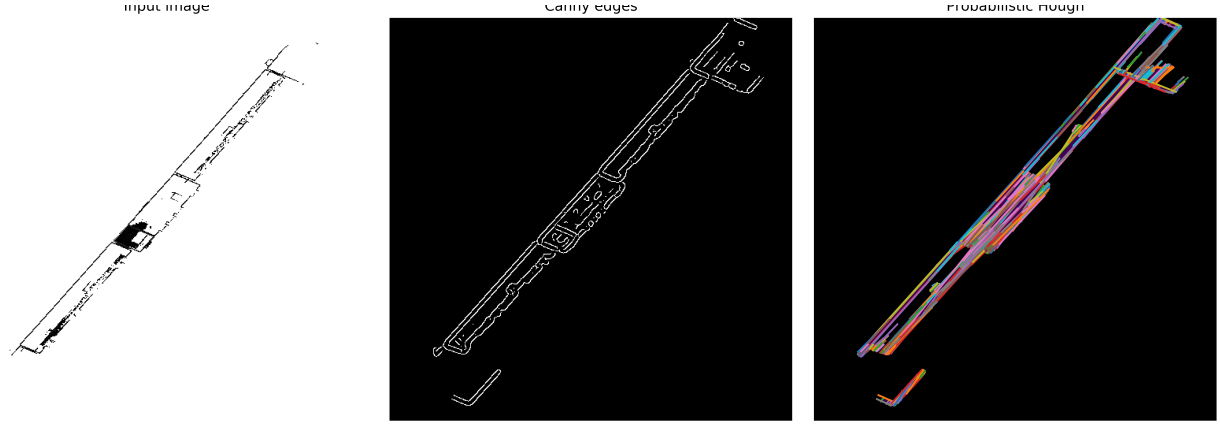

Figure 11. Input image (*left*): Filtered dataset with canny edge extraction (*middle*) parameters $\sigma = 3$, low threshold of 10 and high threshold of 20, line detection using random hough transform(*right*) with threshold of 10, line length of 15 and line gap of 5.

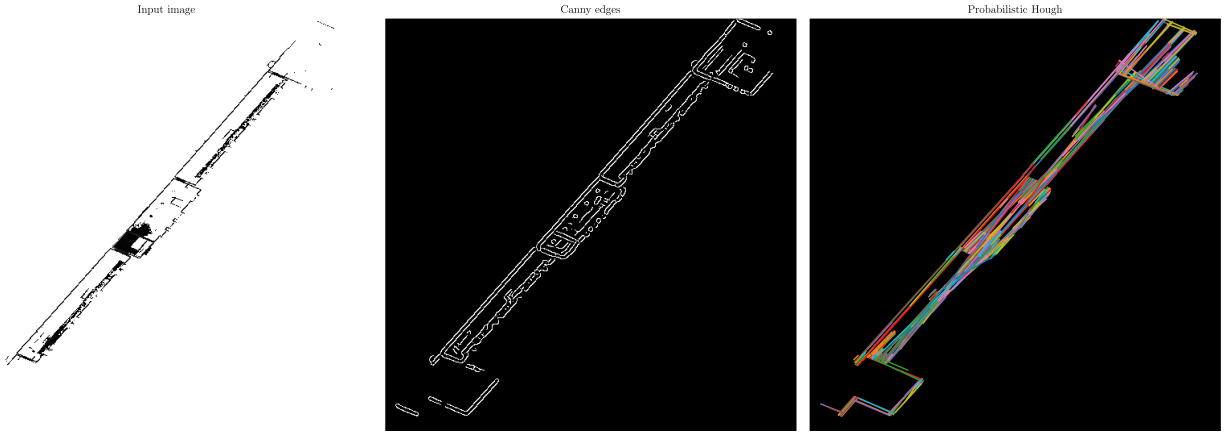

Figure 12. Input image (*left*): Filtered dataset with canny edge extraction (*middle*) parameters $\sigma = 2.5$, low threshold of 10 and high threshold of 30, line detection using random hough transform(*right*) with threshold of 10, line length of 15 and line gap of 20.

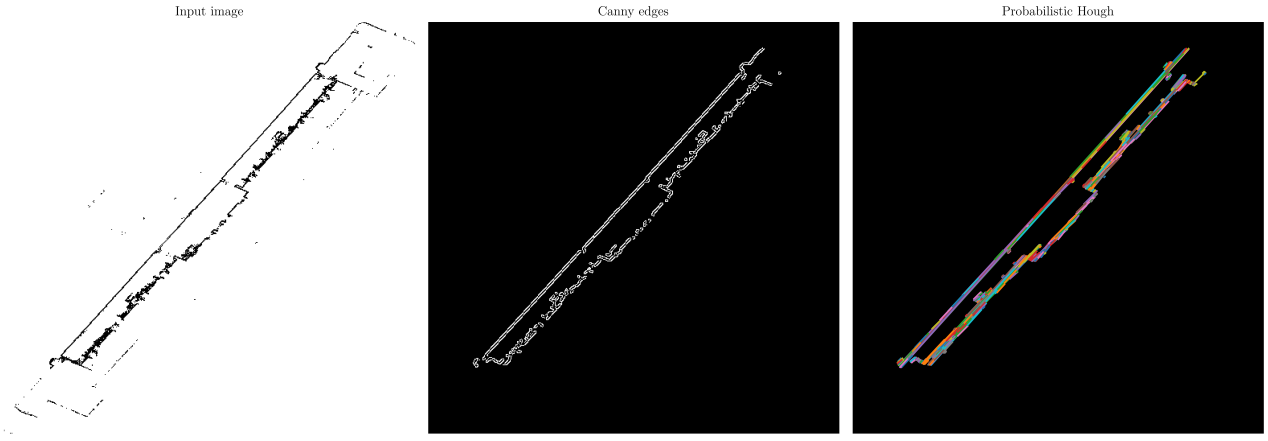

Figure 13. Input image (*left*): Sliced dataset with canny edge extraction (*middle*) parameters $\sigma = 1$, low threshold of 10 and high threshold of 15, line detection using random hough transform(*right*) with threshold of 10, line length of 10 and line gap of 10.

REFERENCES

Babacan, K., Jung, J., Wichmann, A., Jahromi, B., Shahbazi, M., Sohn, G., Kada, M., 2016. TOWARDS OBJECT DRIVEN FLOOR PLAN EXTRACTION FROM LASER POINT CLOUD. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 41.

B.V., H. E., 2018. pptk - Point Processing Toolkit. *PPTK Viewer*. https://github.com/heremaps/pptk.

Canny, J., 2019. Canny edge detector. *Wikipedia*. https://en.wikipedia.org/wiki/Canny_edge_detector.

Cui, Y., Li, Q., Dong, Z., 2019. Structural 3D Reconstruction of Indoor Space for 5G Signal Simulation with Mobile Laser Scanning Point Clouds. *Remote Sensing*, 11(19), 2262.

Flikweert, P., Peters, R., Díaz-Vilariño, L., Voûte, R., Staats, B., 2019. Automatic extraction of a navigation graph intended for IndoorGML from an indoor point cloud. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 4.

Hough, P. V., 1962. Method and means for recognizing complex patterns. US Patent 3,069,654.

Kiryati, N., Eldar, Y., Bruckstein, A., 1991. A probabilistic Hough transform. *Pattern Recognition*, 24(4), 303 - 316. http://www.sciencedirect.com/science/article/pii/003132039190073E.

Liu, C., Wu, J., Furukawa, Y., 2018. Floornet: A unified framework for floorplan reconstruction from 3d scans. 201–217.

Lo, C., Chenb, L., 2012. Structure Line Detection from LIDAR Point Clouds Using Topological Elevation Analysis. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 39, B3.

Okorn, B., Xiong, X., Akinci, B., 2010. Toward Automated Modeling of Floor Plans.

Previtali, M., Barazzetti, L., Brumana, R., Scaioni, M., 2014. Towards automatic indoor reconstruction of cluttered building rooms from point clouds. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 2(5).

Roos, J., Braaksma, Y., 2011. TU Delft Faculty of Architecture. https://miesarch.com/work/1079.

Saalfeld, A., 1999. Topologically Consistent Line Simplification with the Douglas-Peucker Algorithm. *Cartography and Geographic Information Science*, 26(1), 7-18. https://doi.org/10.1559/152304099782424901.

Revised October 2019