On Real-Time LIDAR Data Segmentation and Classification

Dmitriy Korchev[1], Shinko Cheng[2], Yuri Owechko[1], and Kyungnam (Ken) Kim[1]
[1]Information Systems Sciences Lab., HRL Laboratories, LLC, Malibu, CA, USA
[2] Social, Google Inc., Mountain View, CA, USA

Abstract - *We present algorithms for fast segmentation and classification of sparse 3D point clouds from rotating LIDAR sensors used for real-time applications such as autonomous mobile systems. Such systems must continuously process large amounts of data with update rates as high as 10 frames per second which makes complexity and performance of the algorithms very critical. Our approach to the segmentation of large and sparse point clouds is efficient and accurate which frees system resources for implementing other more demanding tasks such as classification. Segmentation is the emphasis of this paper as a necessary important first step for subsequent classification and further processing. We propose methods for segmenting sparse point clouds from rotating LIDAR sensors such as the Velodyne HDL-64E using rectangular and radial grids. The main part of the segmentation is performed on small grid images instead of large point clouds which makes the process computationally fast, simple, and very efficient. The proposed algorithms do not require flat horizontal structure of the ground and they demonstrate stable performance in different urban conditions and scenes for detection of different street objects including pedestrians, cars, and bicyclists.*

Keywords: Real-Time LIDAR, 3D-Segmentation, 3D-Classification

1 Introduction

In this paper we address the problem of segmentation of large sparse LIDAR 3D point clouds in real time for subsequent classification. Advances in LADAR scanner technologies such as the Velodyne HDL-64E allow the development of experimental autonomous mobile systems described in [1,2]. Such systems contain a multi-beam rotating Velodyne LIDAR scanner mounted on top of a vehicle. The LIDAR sensor produces a continuous high data rate stream of 3D points, typically exceeding one million points per second. This stream of points is parsed into frames, where each frame corresponds to one complete rotation of the sensor; the frame rate is close to 10Hz. Each of the 64 lasers in the Velodyne sensor performs a circular scan of the environment, which results in a non-uniform spatial density of the scanned objects. The objects close to the sensor are represented with more dense point clouds vs. far objects that are represented with much sparser point clouds.

While progress has been made, researchers continue to look for new alternative algorithms for segmentation and classification. The goal of segmentation is to parse each separate distinct object in the point clouds for subsequent classification. The main challenges in this process are: high data rate, sparsity of the data, and non-uniform density of the scanned object data. The methods proposed in this paper extend to real-time applications the approach originally developed under DARPA URGENT contract [3,4] for automatic classification of urban objects in large static 3D point clouds collected by aerial and ground LIDARs.

2 Related Work

Efficient real-time object recognition is an important capability for autonomous robots and systems. This drives researchers to make these systems fast, efficient, and accurate. The relevant issues have been analyzed from different perspectives and approaches by various researchers to improve aspects of the system and shed more light on the problems that need to be solved. A number of papers were published in recent years that consider different aspects of real-time segmentation and classification of 3D point clouds [1,2-5-9]. The results presented in [1, 2] demonstrate systems that perform segmentation and classification in real or close to real time. Paper [5] shows good segmentation results using the convexity criterion with promising speed results for real time operation. Paper [6] considered using 3D models to improve classification performance but the overall speed of the system was not sufficient for real-time operations. The research in [7] focused on dense point clouds generated by a flash LIDAR which produces more uniformly dense scans of the objects, unlike spinning LIDARs such as the Velodyne HDL-64E. The approaches in [8] work on both dense and sparse point clouds with close to real-time performance. Another interesting approach to segmentation and classification is presented in [9], the results present Matlab implementation based timing which is promising for real-time operation. Computational complexity of the segmentation is the most significant limitation of the current methods mentioned above, especially for the future low cost embedded applications. The next two sections describe our 3D point cloud segmentation methods based on two different types of grids and a 3D classification algorithm adapted from DARPA URGENT program [4]. We then draw conclusions and compare our results with published results.

3 Segmentation

Our earlier 3D point cloud segmentation method [3, 4] developed under the DARPA URGENT program produced satisfactory segmentation results but it did not meet the real-time requirements that are imposed by the spinning multi-beam LIDARs such as the Velodyne HDL-64E. In this section we present novel methods for real-time segmentation of large sparse point clouds produced by spinning multi-beam LIDAR sensors, requiring processing rates of a million points per second and higher. The novelty of this algorithm is the projection of points onto rectangular or radial grids that allow maintaining point densities in each bin for LIDAR scanning sensors. Segmentation of objects and obstacles is performed by analyzing the minimum and maximum height maps (called Min and Max images) on these grids.

A typical autonomous driving system is based on a spinning multi-laser LIDAR such as the Velodyne HDL-64E which generates point clouds by an array of rotating lasers producing circular scan lines around the sensor. The Velodyne HDL-64E delivers a 360-degree horizontal field of view and 26.8 degree vertical field of view using 64 laser beams. This type of sensor can provide more than 1 million points per second, detecting all directions of environment around it. It has become a popular sensor for building and testing autonomous driving systems. A typical scan pattern of this kind of LIDAR as seen from above is shown in Figure 1. The vertical array of lasers rotates with a constant speed and produces a fixed number of scan lines (actually circles on the horizontal plane) per spin. That results in uneven spatial density of the scan lines, in particular, the scan line density decreases with the distance from the sensor. Since each scan line contains the same number of points, the density of the points within the scan lines also decreases with the distance from the sensor. These properties of the LIDAR lead to highly sparse point clouds at longer ranges compared to shorter ranges.

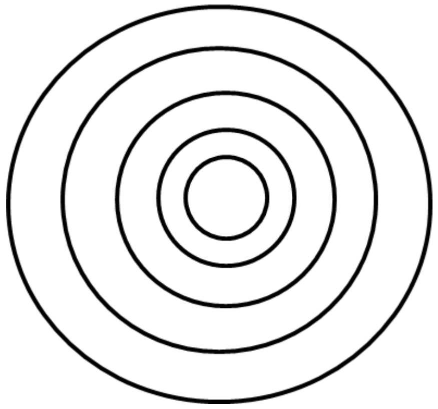


Figure 1 A typical Velodyne scan pattern shown in horizontal plane. Only five laser lines are shown.

We investigated the segmentation performance of two types of sampling grids: rectangular and radial. The initial processing of each grid is very similar. Every complete spin of the lasers produces a complete point cloud or frame of the scene. Each frame is projected into the grid along the z-axis, and is represented by minimum and maximum height maps called *Min image* and *Max image*, respectively. The Min and Max images have the same size defined by the type, size and parameters of the grid. A separate structure of the same size as the images is used to hold the associated points for each grid cell. This association is necessary to perform the reverse lookup to efficiently extract point cloud segments from these images. The processing of each grid is based on the general procedure shown in Figure 2.

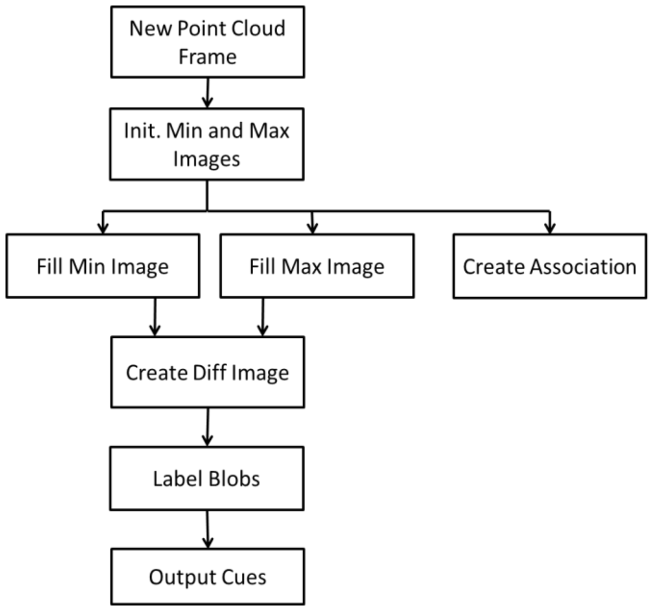


Figure 2 Block diagram of the grid processing of a single frame

The Min and Max images are 8-bit images representing 256 height gradations. Before processing a point cloud frame, all pixels of these images are set to 255 for the Min image and to 0 for the Max image.

3.1 Rectangular Grid

Let's define the max distance from the sensor as D_{max} and d as the lateral size of the grid cell. The dimensions of the Max and Min images in terms of rows H and columns W will be

$$W = H = \frac{2 * D_{max}}{d} + 1.$$

Parameter h defines the vertical resolution of the grid. Typical values for the vertical and lateral resolutions for automotive applications are: $h = 10cm$, $d = 25 - 35cm$. The block diagram shown in Figure 3 describes the process of filling the

Min and Max images and associating the point with a grid-cell.

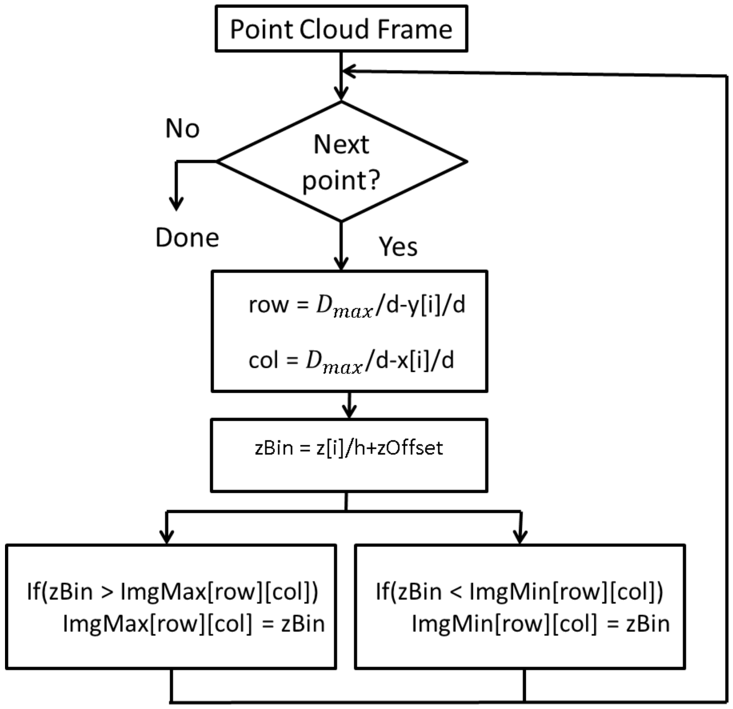
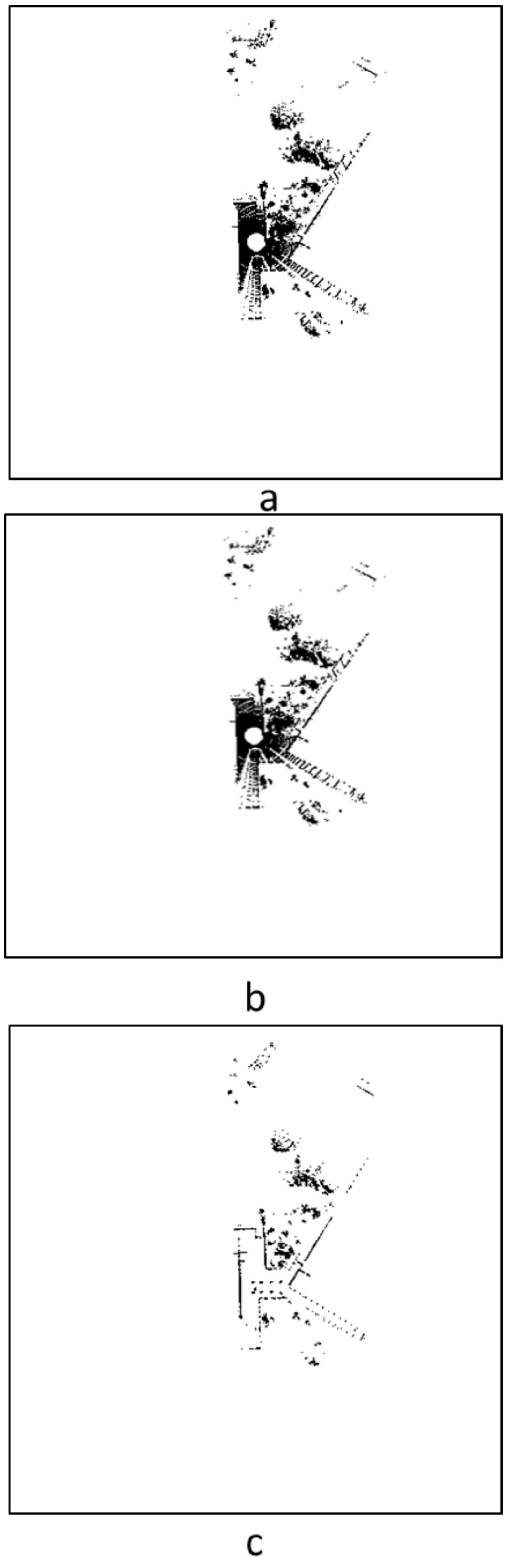


Figure 3 Bock diagram of filling the Min and Max images for the rectangular grid

Examples of Min, Max images for the rectangular grid are shown in Figure 4 (a) and (b), respectively. We then process the Min and Max images to obtain the Diff image, non-zero pixels of which contain elevated cells of the grid. The process of finding these locally elevated objects does not require a flat ground plane due to the procedure defined below. The Diff image is created by the procedure based on the small sliding windows of size MxM running in parallel in both Min and Max images. Typical size M for this window is from 1 to 5 for the grid cell sizes presented in this paper. The size of the window should be adjusted accordingly for different sizes of the grid cells to be able to capture local ground. The following steps describe the procedure of filling the Diff image:

1. For location of the window <i,j> find the min pixel value Pmin in the Min image.
2. Mark Diff image location <I,j> as 255 if abs(Pmin-Pmax(I,j) >= T and the cell is not empty
3. Move window to the next location.

The resulting Diff image is shown in Figure 4c. The Diff image is then processed with 8-connected component analysis that will label the blobs as shown in Figure 4d.



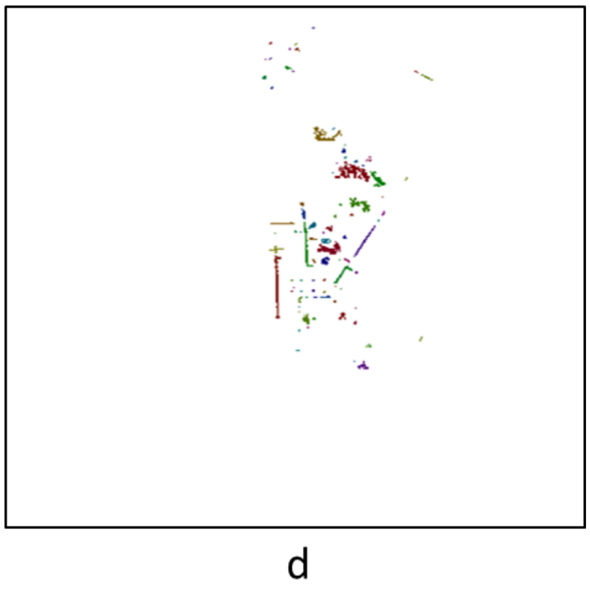


Figure 4 Rectangular grid: (a) - Min image, (b)- Max image, (c) - Diff image, (d) - Blob image. The sensor is located in the center of the image. Intensities of images (b) and (c) are inverted for better clarity.

The point cloud frame and the segmentation results based on the procedure described above for the rectangular grid are shown in Figure 5. Each segmented cue is colored with distinct random color. The ground points and points that do not belong to any other object are black.

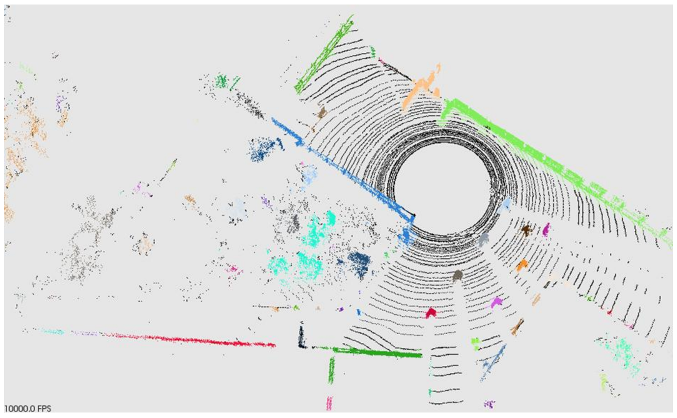


Figure 5 Segmentation results of our method with the rectangular grid, the connected segments are randomly colored; the ground points have black color. We can see that our method of segmentation handles different object correctly, including thin rail surrounding the patio.

We found that value of threshold *T=1* produces good result for a variety of scenes and objects. After that, each blob is processed to generate the segmented point cloud. This process is accomplished by extracting the pixels that belong to the same blob and collecting the indices of the points in the original point cloud.

3.2 Radial Grid

A part of the radial grid is shown in Figure 6. This grid is better aligned with Velodyne scan pattern shown in Figure 1 and it produces less fragmentation of the segments.

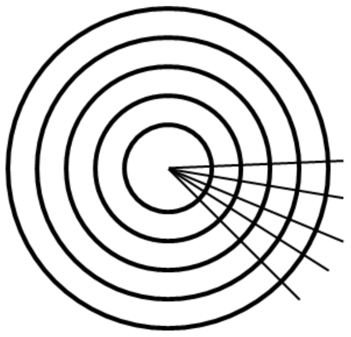


Figure 6 Fragment of radial grid on horizontal plane

The width *W* and height *H* of the Min and Max images are determined as

$$W = 360/ResDeg,$$

$$H = (MaxRadius\text{-}MinRadius)/RangeRes.$$

Where, *ResDeg is the* angular resolution and *RangeRes* is the range resolution. The block diagram of filling the Min and Max images for the radial grid is shown in Figure 7. The Diff and Blob images and the association of the points are created the same way as it was described for the rectangular grid.

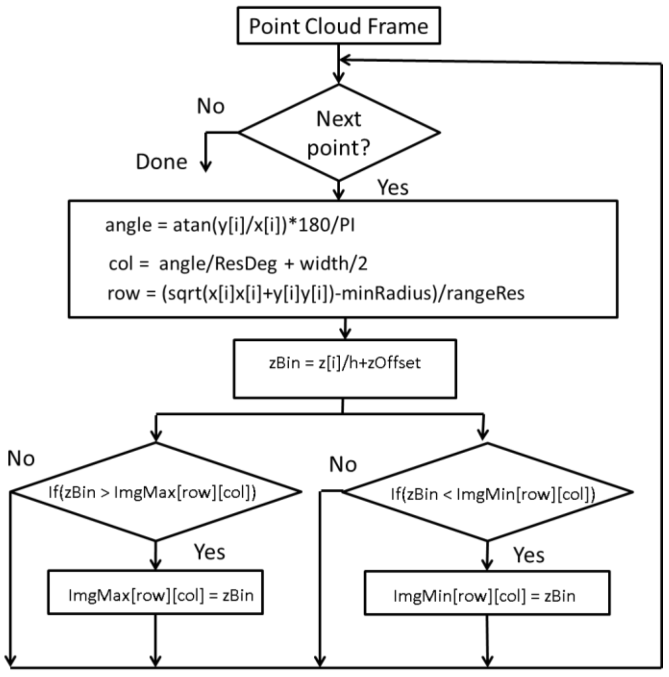

Point Cloud Frame
No
Next point?
Done
Yes
angle = atan(y[i]/x[i])*180/PI
col = angle/ResDeg + width/2
row = (sqrt(x[i]x[i]+y[i]y[i])-minRadius)/rangeRes
zBin = z[i]/h+zOffset
No
If(zBin > ImgMax[row][col])
Yes
ImgMax[row][col] = zBin
No
If(zBin < ImgMin[row][col])
Yes
ImgMin[row][col] = zBin

Figure 7 Block diagram of filling the Min and Max images for the radial grid

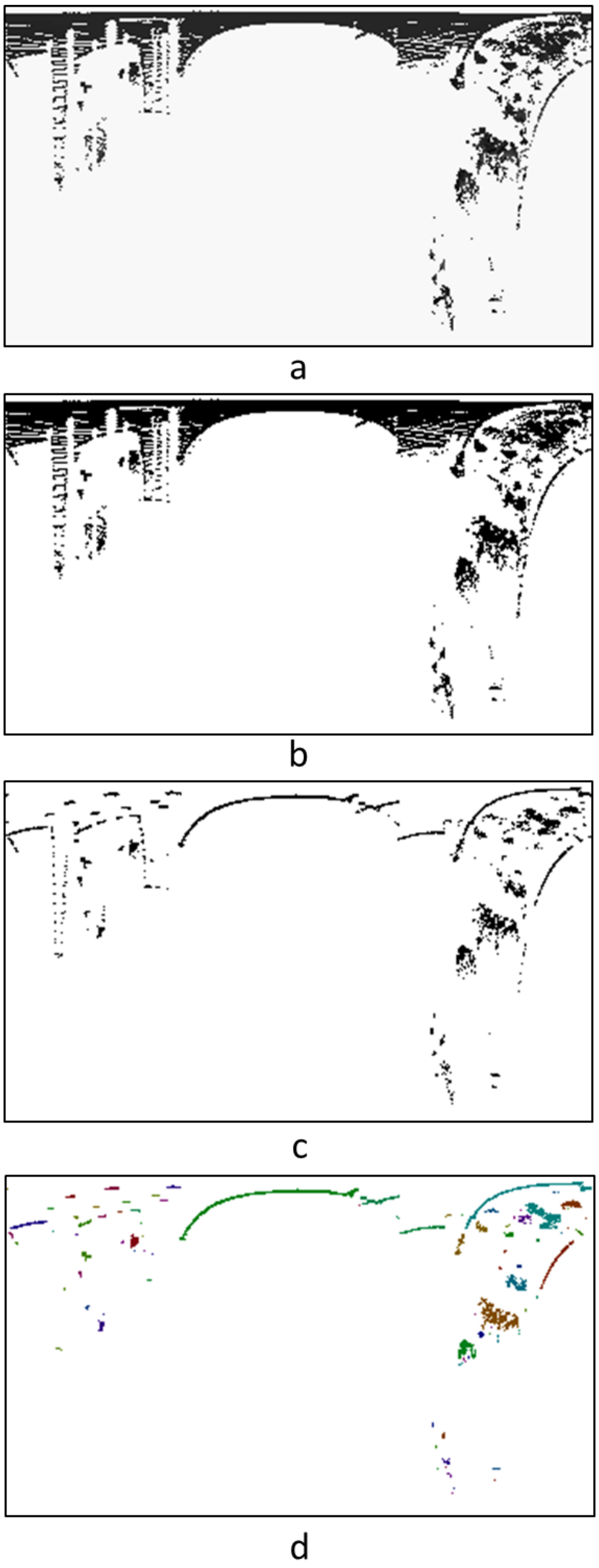


Figure 8 Radial grid: (a) - Min image, (b) - Max image, and (c) - Diff image, (d) Blob image. Intensities of images (b) and (c) are inverted for better clarity.

Overall segmentation of the data is less for the radial grid compared to the rectangular one. Horizontal axes define azimuth direction and vertical axis - the distance from the sensor. The sensor is located in the middle of the top row or above it depending on parameter that defines the minimum distance from the sensor. Forward direction points downward from the middle of the top row. Backward direction points downward at the left and right edges of the image. The objects that are directly behind the sensor need to be stitched because their parts appear on the right and left edges of the images.

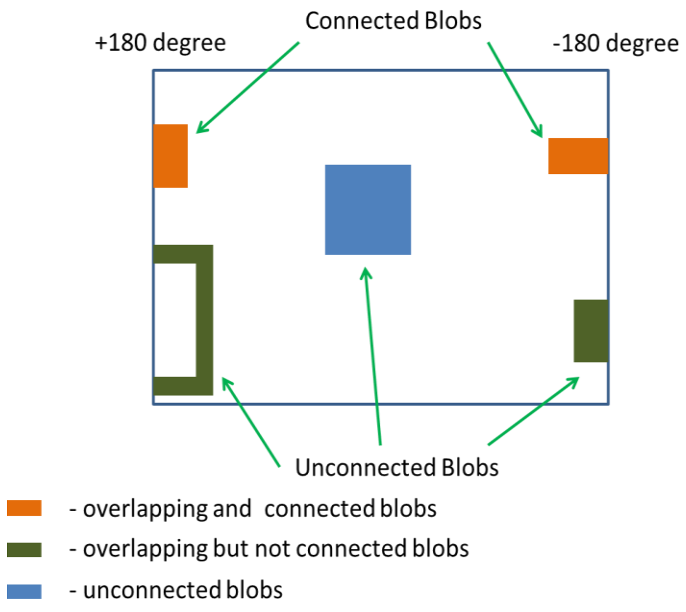


Figure 9 Demonstration of the stitching for the radial grids. Only "connected" blobs need to be merged, unconnected and internal blobs do not need merging.

The radial grid Min, Max, Diff and Blob images are shown in Figure 8. When radial grid is used, we need to stich some of the blobs at the right and left edges (see more details in the caption under Figure 8). These edges represent locations behind the sensor and are split by the line representing ±180 degree in azimuth. The blobs that need to be stitched are located on the left (col=0) and right (col=N-1) sides of the Blob image. Only objects that got cut by the ±180 degree azimuth need to be stitched. In the first step of the stitching process we select blobs that connect to the left and right edges of the image Figure 8 (d). After that we find the blobs that have overlapping vertical pixel coordinates for pixels belonging to the first and last columns, left and right of the image, correspondingly. These blobs are marked "connected" as it is shown in Figure 9 and they are merged into one blob representing a single object behind the sensor. This process is applied to all border blobs to correctly represent the objects located directly behind the sensor.

An example of a point cloud frame and segmentation with the radial grid is shown in Figure 10.

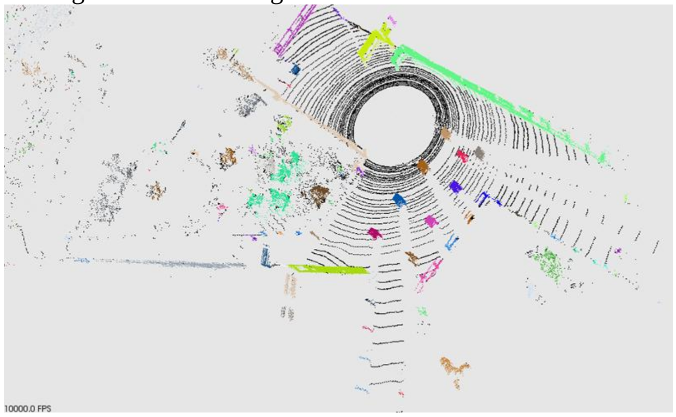


Figure 10 Segmentation results of our method with the radial grid, the connected segments are randomly colored; the ground points have black color. We can see that our method of segmentation handles different object correctly, including thin rail surrounding the patio.

The radial grid requires 20-30% fewer cells compared to the rectilinear one, which leads to faster processing speed. It also produces less fragmentation of the objects. An illustration of the fragmentation for a typical sequence of frames acquired while urban driving is shown in Figure 11. The chart shows that the average number of fragments is 12-13% fewer for the radial grid.

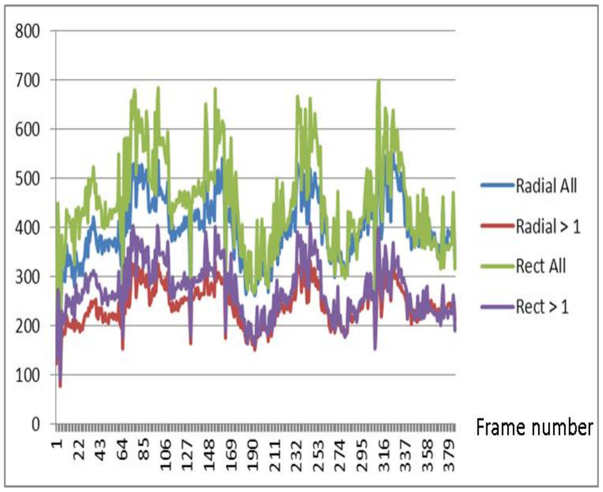


Figure 11 Charts showing number of fragments in each frame for a typical urban sequence for the radial and rectangular grids. Each type of grid has two charts: one for the total number of fragments in a frame and one for the number of fragments that are greater than one grid cell.

Table 1 Comparison of the fragmentation for rectangular and radial grids. The table shows that overall fragmentation of the segments for the radial grid 12-13% fewer than for the rectangular grid.

Blobs type	Average number of blobs
Radial (all blobs)	398.6
Radial (blobs bigger than 1 cell)	242.8
Rectangular (all blobs)	451.0
Rectangular (blobs bigger than 1cell)	271.8

4 Integration of 3D Segmentation with Classification

The block diagram of the segmentation/classification system we developed is shown in Figure 12. It consists of the Velodyne LIDAR, a converter from pcap format to XYZ point clouds, the 3D segmentation module described in the previous chapter, and the 3D classification module that is described below.

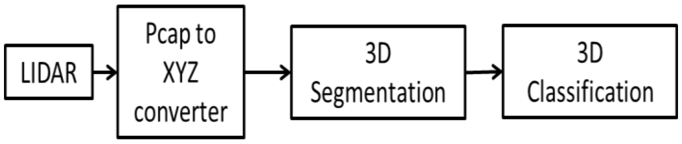


Figure 12 Block diagram of the developed system.

We used a modified 3D classifier [9] developed under the DARPA URGENT program for 17 urban objects. The core feature set for this classifier is based on size and rotation invariant volumetric features [3]. The classifier was developed to process dense point clouds acquired by aerial and ground LIDARs. More info and results for this classifier can be found in [3].
In our current work, we defined four classes which are the most relevant to autonomous mobile systems. These classes are: car, pedestrian, bicyclist, and background. We used 75 minimum points as the limit of the number of points in the cue as suggested in [2]. To validate the performance of the classifier we used the data set in [2] which contains significant numbers of labeled objects for car, pedestrian, bicyclist, and background. Overall performance of the classifier is presented in the table, Table 2. Each column in the table presents the result of the classifier trained on cues with minimum number of points 500, 250, 150, and 75. We used 5k examples for car, pedestrian, bicyclist, and 20k examples for background class. The table represents the performance of the classifiers on validation data sets that do not contain any training examples. The accuracy for pedestrian and bicyclist degrades about 5% with the reduction of minimum number of points in a cue from 500 to 75. The results suggest that the overall performance of the classifier achieved for these four classes is close to the state of the art results [2].

Table 2 Classification accuracy for car, pedestrian, bicyclist, and background. Each column represents classification accuracy for 500, 250, 150, and 75 minimum points in the cue.

Min number of points	500	250	150	75
Car		0.96	0.95	0.91
Pedestrian	0.93	0.91	0.9	0.88
Bicyclist	0.95	0.93	0.93	0.9
Background	0.98	0.98	0.98	0.97

5 Segmentation Results

The goal of segmentation and classification of 3D data is to achieve accurate performance and in real time which is the necessary part of an autonomous mobile system.

Due to unavailability of labeled data sets containing complete scans and difficulty of manual labeling of the data, we evaluated the performance of our system on our data qualitatively. The examples of the segmentations for rectangular and radial grids are presented in Figure 5 and Figure 10, respectively. We used different urban scenes with flat and inclined ground to evaluate the segmentation. We demonstrated that our methods handle variety of objects with different shapes and sizes correctly. The proposed methods of segmentation also handle thin objects like rails dividing the roads and sidewalks as well as roofs of the cars correctly; this importance was emphasized in [1]. Overall segmentation and classification results on Velodyne data were good with an acceptable amount of under and over segmentations. The time taken by the segmentation implemented on a single thread running on HP workstation Z400 was 30-40ms per frame depending on the grid parameters settings. We did not observe any significant fluctuations of processing time for different scenes and objects. This speed is more than enough to do the segmentation of Velodyne HDL-64E data in real time.

We also evaluated the performance of the 3D classifier on a labeled public data set from [2].The results of this evaluation show that the performance of our modified URGENT classifier is very close to other state of the art results.

6 Conclusions and Future Work

We developed a new 3D segmentation and classification framework that processes high volumes of LIDAR data in real time. The proposed segmentation algorithms can be applied to a variety of different applications and scenes; in particular, they can be used in autonomous mobile systems. We achieved good segmentation and classification quality and real-time performance of the system due to the novel approach to the segmentation of large sparse 3D point clouds. The core of the approach is based on processing of smaller images instead of large point cloud data. The methods proposed do not require a flat ground plane and they reliably handle a variety of complex urban environments and objects of interests.

Our future work will consist of integrating additional features such as tracking of 3D objects and fusion between LIDAR and EO sensors to improve overall system performance.

7 References

[1] M. Himmelsbach, F. Hundelshausen, H. Wuensche. Fast segmentation of 3D point clouds for ground vehicles. *Intelligent Vehicles Symposium (IV)* (pp. 560 - 565). IEEE, 2010

[2] A. Teichman, J Levinson, S.Thrun. Towards 3D object recognition via classification of arbitrary object tracks . *Internationa Conference on Robotics and Automation (ICRA)* (pp. 4034 - 4041). IEEE, 2011.

[3] Y. Owechko, S. Medasani, T. Korah. Automatic Recognition of Diverse 3-D Objects and Analysis of Large Urban Scenes Using Ground and Aerial LIDAR Sensors. Conference on Lasers and Electro-Optics. San Jose: Optical Society of America, 2010.

[4] T. Korah, S. Medasani, Y. Owechko. Strip Histogram Grid for efficient LIDAR segmentation from urban environments. *Conference on Computer Vision and Pattern Recognition Workshops, (CVPRW)* (pp. 74-81). IEEE, 2011.

[5] F. Moosmann, O. Pink, C. Stiller. Segmentation of 3D Lidar Data in non-flat Urban Environments using a Local Convexity Criterion. Intelligent Vehicles Symposium, pp.215-229, IEEE, 2009.

[6] K. Lai, D. Fox. 3D Laser Scan Classification Using Web Data and Domain Adaptation. International Journal of Robotics Research, Volume 29 Issue 8, pp. 1019-1037, 2010

[7] J. Aue, D. Langer, B. Muller-Bessler, B. Huhnke. Efficient Segmentation of 3D LIDAR Point Clouds Handling Partial Occlusion. Intelligent Vehicles Symposium IV, pp. 423-428, 2011

[8] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, A. Frenkel. On the Segmentation of 3D LIDAR Point Clouds. International Conference on Robotics and Automation, pp.2798-2805, IEEE, 2011.

[9] S.-M. Lee, J. J. Im, B.-H. Lee, A. Leonessa, A. Kurdila. A Real-Time Grid Map Generation and Object Classification for Ground-Based 3D LIDAR Data using Image Analysis Techniques. International Conference on Image Processing, ICIP, pp. 2253-2256, IEEE, 2010.