2020 IEEE Intelligent Vehicles Symposium (IV)
October 20-23, 2020. Las Vegas, USA

Estimation of 2D Bounding Box Orientation with Convex-Hull Points – A Quantitative Evaluation on Accuracy and Efficiency

Yang Liu, Bingbing Liu and Hongbo Zhang

Abstract— **Estimating the bounding box from an object point cloud is an essential task in autonomous driving with LiDAR/laser sensors. We present an efficient bounding box estimation method that can be applied on 2D bird's-eye view (BEV) LiDAR points to generate the bounding box geometry including length, width and orientation. Given a set of 2D points, the method utilizes their convex-hull points to calculate a small set of candidate directions of the box yaw orientation, and therefore reduces the searching space – usually a fine partition of an angle range (e.g. [0, $\pi/2$)) as in the previous solutions – to find the optimal angle. To further improve the efficiency, we investigate the techniques of controlling the number of convex-hull points, by both applying approximate collinearity condition and downsampling the raw point cloud to a smaller size. We provide comprehensive analysis on both accuracy and efficiency of the proposed method on the KITTI 3D object dataset. The results show that without obviously sacrificing the accuracy, the method, especially when using approximate convex-hull points, can significantly improve the time of estimating the bounding box orientation by almost one order of magnitude.**

I. INTRODUCTION

The rapid development of autonomous driving (AD) in the recent years has been drastically driving the research of 3D object detection from point cloud obtained by LiDAR sensors. There has been an increasing amount of work addressing the LiDAR-only solution to environment perception, including STD [1], PointPillars [2], PointRCNN [3], SECOND [4], PIXOR [5], VoxelNet [6], Complex-YOLO [7], among others. While most of these approaches provide an end-to-end solution to 3D object detection based on a 3D convolutional neural network, another type of LiDAR perception is to process the raw point cloud in a pipeline consisting of a series of tasks including, for example, segmentation, clustering, bounding box estimation, etc., to output a set of object proposals as in a neural network [8], [9], [10]. One main advantage of the pipeline-based methods is that they do not need prior information of the environment for training. In addition, the vertical dimension can be marginalized in some tasks including point cloud segmentation and object tracking in 2D, to generate a more compact representation of the environment features using, for example, 2D BEV points. The work in this paper focuses on estimating the object bounding box with BEV LiDAR points in a LiDAR perception pipeline described above.

Given a set of 2D points, the proposed method first finds a set of convex-hull points, it then calculates the line angle (with x axis) formed by each pair of points in this set as a candidate orientation of the bounding box. A score can be associated with this angle by evaluating the object points with criteria such as area and closeness [15]. Finally, the angle with the optimal score will be selected as the bounding box orientation and the corresponding box length and width can be also obtained. Figure 1 shows the four steps of the method. Note that a non-learning method usually fits a bounding box that will tightly embrace all the 2D points without predicting the actual size of the object. As a result, the estimated length and width may not be accurate in some circumstances such as occlusion, where the object points are partially missing. In this paper, we focus on evaluating the orientation accuracy of the proposed method and omit the size information. In an autonomous driving system, the bounding box size can be usually compensated and corrected within a few frames in the following modules such as tracking or data association, conditioned on the accurate estimation of the orientation.

We also investigate the possible techniques to futher improve the efficiency of extracting the bounding box when an object contains thousands of points, without obviously sacrificing the accuracy. Particularly, we show that by applying approximate collinearity judgement in the selection of convex-hull points, and downsampling the point cloud to a smaller size, we are able to reduce the number of convex-hull points while retaining the optimal or sub-optimal orientation candidate. The main contributions of the paper include:

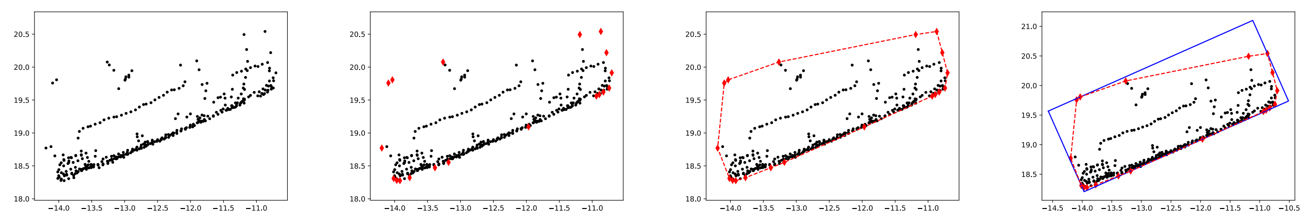
- We present an efficient bounding box estimation method with convex-hull points.
- Different techniques are used to improve the efficiency of the original method.
- Thorough quantitative analysis on both accuracy and efficiency of the method is provided to serve as a guideline of applying it in an AD system.

The rest of the paper is organized as following: Section II concisely discusses the background, with a focus on 2D L-shape fitting which is the most relevant to this work. The proposed method is illustrated in Section III with techniques of improving the efficiency. We provide comprehensive experimental validation on the KITTI 3D object dataset in Section IV, and briefly conclude the paper in Section V.

II. RELATED WORK

Bounding box regression is an essential step in an end-to-end solution to 3D object detection using neural networks. Since our method only addresses the 2D case without depending on any learning-based approaches, we omit the vast literature review of the mainstream neural network solutions, but focus on the relevant work of L-shape fitting of 2D laser

The authors are with Huawei Noah's Ark Lab, Canada. {yang.liu6, liu.bingbing, zhanghongbo888}@huawei.com



(a) Input (b) Convex-hull points (c) Candidate directions (d) Bounding box

Fig. 1: Illustration of the method. (a): a set of 2D BEV points as the input. (b): convex-hull points shown in diamond. (c): candidate directions shown in dashed line by connecting a pair of points. Note that two non-adjacent points can also form a candidate direction (not shown in the figure for clarity). (d): bounding box shown in solid line with the optimal direction.

scanner data (or BEV LiDAR point cloud) in modeling a vehicle [11], [12].

A standard least-square method was used in [11], an early example to fit both line and corner features in a rectangle model, with a weighting strategy to remove outliers. However, details were not mentioned in the method. In [12], “virtual scan" was used to reorganize the LiDAR point cloud for compact feature representation. A probabilistic measurement model was used to describe the occupancy of each grid along the virtual ray, and vehicle fitting was then achieved by calculating the scan differencing. A potential issue of virtual scan is that the information may be lost by quantizing the points. Line fitting problem was solved in [13] by formulating the points in a matrix and optimizing the parameters that correspond to the line coefficients. The method requires an ordered set of points with high-order polynomial time complexity to find the optimal solution. In [14], a K-L transform was applied to the laser points to obtain the two axes that ideally correspond to the edges of the object. More recently, three evaluation criteria were proposed in [15] within an optimization framework to generate a bounding box that can embrace all the points. Instead of solving a complex optimization problem to find the best orientation, the method searches in a fine partition of a range $[0,\pi/2)$ and selects the angle that optmizes the evaluation criteria. In [16], the points were iteratively clustered to two orthogonal segments using their ordered information. A learning-based method was proposed in [17] to predict the bounding box geometry with a neural network.

The method described in [18] was the most similar to ours. It uses convex-hull points to calculate the box orientation based on multiple evaluation criteria. However, the method only considers the lines formed by adjacent points and ignores all the other pairs, possibly for efficiency consideration. Our method instead, retains the full combination of point pairs to ensure accuracy, and improves efficiency by controlling the number of convex-hull points through approximate collinearity determination and point cloud downsampling. In addition, we provide detailed analysis on the accuracy and efficiency of using convex-hull points to estimate a 2D bounding box, which is not fully explored in [18]. This quantitative analysis is important because it not only provides techniques of improving the algorithm efficiency, but also statistically characterizes the relationship or tradeoff between accuracy and efficiency, and therefore generates the guideline of using the method in a real AD system.

III. Bounding Box Estimation with Convex-Hull Points

The proposed method consists of a few essential steps, including convex hull extraction and score calculation that requires one or multiple metrics to evaluate a candidate orientation. From this perspective, it is well supported by the current relevant research and can benefit from the existing results. In this section, we will discuss the detailed implementation of the algorithm, as well as its fundamentals.

Given a set of n points $\{P_i(x_i,y_i) \in \mathbb{R}^d | i = 1,2,\dots,n\}$, where $d = 2$ in our case and (x_i,y_i) represent the 2D coordinates, finding the convex-hull points $C \subseteq P$ is a well-studied problem with many sophisticated solutions including, e.g., gift wrapping algorithm [19], Graham scan [20], Chan’s algorithm [21], among others. For consideration of both simplicity and efficiency, we choose Graham scan in our method to generate a set of convex-hull points, with time complexity $O(n\log n)$. The algorithm is briefly summarized in Algorithm 1. In Algorithm 1, $\texttt{ccw} > 0$ if three points make a counter-clockwise turn, clockwise if $\texttt{ccw} < 0$, and collinear if $\texttt{ccw} = 0$. Basically, the algorithm starts with the lowest point P_0 (selects the leftmost one if there are multiple points with the same smallest y value) and iterates through the sorted point set based on the polar angle relative to P_0. By using a stack, only the points constructing a counter-clockwise (left) turn with the top two points in the stack will be pushed and retained. The final convex-hull points are the ones that are kept in the stack. Readers who are interested in the details of the algorithm can resort to [20].

Instead of using function interfaces in the open source library such as OpenCV, we implemented Graham scan based on Algorithm 1. One advantage of the manual implementation is that it provides us the flexibility to modify and improve the details, and investigate the key factors in the algorithm that can potentially impact the performance and efficiency. For example, one does not need to explicitly calculate the angle formed by three points in line 5 (sorting

Algorithm 1: Convex-Hull Points Extraction

```
1  Function ConvexHull(P)
2      // Find the lowest point
3      $P_0 \leftarrow$ lowest(P);
4      // Sort points by polar angle with $P_0$
5      $P \leftarrow$ sort($P_0$,P);
6      // If collinear, keep the farthest one
7      $m \leftarrow 1$;
8      for $i \leftarrow 1$ to $n$ do
9          while $i < n-1$ and ccw($P_0,P_i,P_{i+1}$) $= 0$ do
10             $i \leftarrow i+1$;
11         end
12         $P_m \leftarrow P_i$;
13         $m \leftarrow m+1$;
14     end
15     if $m < 3$ then            // Not sufficient points
16         return $\varnothing$;
17     end
18     Push $P_0, P_1, P_2$ to stack;
19     // Scan the points
20     for $i \leftarrow 3$ to $m$ do
21         while stack.size $\geq 2$ and
            ccw(stack.next_top, stack.top, $P_i$) $\leq 0$ do
22             Pop stack;
23         end
24         Push $P_i$ to stack;
25     end
26     return stack.elements;
```

based on polar angle), 9 and 21. Instead, using any monotonic function of the angle can be more efficient. Here we use the sign of the cross product which is defined in Eq. (1) to determine relative angles:

$$\mathbf{a} \times \mathbf{b} = \|\mathbf{a}\|\|\mathbf{b}\| \sin(\theta)\mathbf{n} \tag{1}$$

where θ is the angle between two vectors **a** and **b**, $\|\cdot\|$ means magnitude and **n** is a unit vector perpendicular to the plane containing $\|\mathbf{a}\|$ and $\|\mathbf{b}\|$ in the direction given by the right-hand rule. In our case, for three points $P_1(x_1,y_1)$, $P_2(x_2,y_2)$ and $P_3(x_3,y_3)$, the cross product between $\overrightarrow{P_1P_2}$ and $\overrightarrow{P_2P_3}$ is calculated as:

$$\overrightarrow{P_1P_2} \times \overrightarrow{P_2P_3} = (x_2-x_1)(y_3-y_2)-(y_2-y_1)(x_3-x_2) \tag{2}$$

and their magnitudes are:

$$\|\overrightarrow{P_1P_2}\| = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2} \tag{3}$$

$$\|\overrightarrow{P_2P_3}\| = \sqrt{(x_3-x_2)^2+(y_3-y_2)^2} \tag{4}$$

The sinusoid value of $\angle P_1P_2P_3$, according to Eq. (1) is:

$$\sin\angle P_1P_2P_3 = \frac{\overrightarrow{P_1P_2} \times \overrightarrow{P_2P_3}}{\|\overrightarrow{P_1P_2}\| \cdot \|\overrightarrow{P_2P_3}\|} \tag{5}$$

If $\sin\angle P_1P_2P_3 = 0$, the points are collinear; if it is positive, the three points constitute a counter-clockwise orientation, otherwise a clockwise orientation. Eq. (5) can be used to implement function ccw in Algorithm 1. Particularly, we can modify ccw $= 0$ in line 9 to $\|\sin\angle P_0P_iP_{i+1}\| \leq \tau$, where τ is a given small positive threshold to control the collinearity determination. Similarly in line 21, the same function can be changed to $\sin\angle P_{t_2}P_{t_1}P_i \leq \tau$, where P_{t_2} and P_{t_1} are the next-to-top and top points in the stack respectively. By removing the points that are approximately collinear, we are able to obtain fewer convex-hull points and therefore reduce the searching space of candidate orientation to improve the efficiency. The performance will be validated in the experimental results in Section IV.

The bounding box estimation algorithm is summarized in Algorithm 2. For each candidate direction formed by a pair of points in C, we calculate the evaluation criterion such as closeness, and update the size and orientation associated with the current calculation if a better score is obtained. In this case, a smaller closeness means a better orientation [15].

The efficiency of Algorithm 2 mainly depends on both the time of running Algorithm 1 and its output. There are $n(n-1)/2$ operations in the nested loops. Therefore, in order for the method to be efficient, the convex-hull points need to be well controlled to a reasonable size (e.g. lower than 10) without impacting the performance too much. On the other hand, the size of the raw point cloud usually has a positive correlation with the number of convex-hull points. These facts become our basis of improving the method.

IV. EXPERIMENTS

We validated the proposed method on the KITTI 3D benchmark [22]. Implementation was in C++ and experiments were running on a regular lab computer.

Algorithm 2: Bounding Box Estimation

```
    Input  : Set P, consisting of n 2D points
    Output: $\theta$, length and width
1   $S$ = INT_MAX;
2   if $|P| > N_{max}$ then        // $N_{max}$ is a given threshold
3       $P \leftarrow$ downsample(P);
4   end
5   $C \leftarrow$ ConvexHull(P);
6   if $|C| = 0$ then              // No convex hull generated
7       return;
8   end
9   for $i \leftarrow 0$ to $|C|-1$ do
10      for $j \leftarrow i+1$ to $|C|$ do
11          $\theta_{cur}$ = atan2($y_j - y_i, x_j - x_i$);
12          $S_{cur}, l_{cur}, w_{cur}$ = closeness(P, $\theta_{cur}$);
13          if $S_{cur} < S$ then
14              $\theta = \theta_{cur}$;
15              length $= l_{cur}$;
16              width $= w_{cur}$;
17          end
18      end
19  end
```

TABLE I: Number of test cases with a given number of LiDAR points

≥ 3	≥ 20	≥ 40	≥ 60	≥ 80	≥ 100	≥ 120	≥ 140	≥ 160	≥ 180	≥ 200	≥ 1000
28,190	21,902	18,376	16,080	14,441	13,317	12,352	11,565	10,860	10,215	9,683	3,095

A. *Data Preprocessing*

The KITTI 3D object dataset has 7,481 image frames and their corresponding LiDAR point clouds, each of which contains one or multiple objects. The data are well annotated in 3D format in camera frame with accurate sensor calibration. Therefore, we first transformed the point cloud in each LiDAR frame to the camera coordinate system and then extracted all the points within every ground truth bounding box according to the annotation of that frame. In our evaluation, we focused on the *car* category, not only because it is the most common and important class and has been emphasized in most AD research, but also due to its better L-shape (or rectangular) property that is more suitable to the validation of the method relative to the other classes. There are totally 28,742 car objects in the dataset. However, we only considered test cases with 3 points or more so that the algorithms can apply. Table I summarizes the number of test cases with a given number of LiDAR points, e.g., the second column means that there are 21,902 cases with at least 20 points.

B. *Evaluation Method*

As mentioned in Section I, in this work, we focused on evaluating the orientation accuracy instead of the size, even though the bounding box estimation algorithm can provide length and width. The reason is that partial points may still generate correct orientation. The size however, may be more likely influenced by the missing points. To obtain the difference between the estimated orientation and the ground truth, we first converted the ground truth value from $[-\pi, \pi]$ to $(-\pi/2, \pi/2]$, and then calculated the following errors:

$$err_1 = |\theta - \theta_{true}| \tag{6}$$

$$err_2 = |\pi - err_1| \tag{7}$$

$$err_3 = |\pi/2 - err_1| \tag{8}$$

where θ_{true} is the ground truth. The real error is defined as:

$$err = \min(err_1, err_2, err_3) \tag{9}$$

The reason of using Eq. (9) is to account for the possible length and width switch when LiDAR points are partially observed. In other words, 0, $\pi/2$ and π are considered the same in terms of evaluating orientation difference in our study.

C. *Accuracy*

First, we are interested in understanding the accuracy of the proposed method. In order to achieve the evaluation on accuracy, we extracted the convex-hull points for all the 28,190 test cases using Algorithm 1, with strict condition on collinearity determination, i.e., $\tau = 0$ (see Section III), and calculated their bounding box orientations with Algorithm 2. We entitled this method "Accurate Convex Hull" since it guarantees to generate the correct convex-hull points. On the other hand, the baseline competitor is the method used in [15], where an exhaustive search of orientation in the range of $[0, \pi/2)$ was used with closeness as the criterion to find the optimal orientation. We used a step size of 0.01 radians in our implementation, resulting in 157 trials to complete the search.

Figure 2 shows the accuracy comparison between two methods. The x axis specifies the number of test cases with a given number of LiDAR points (see Table I for detail, the number means equivalent or higher), while the y axis shows the average error in degree. As the number of points increases, both methods will have better estimation of bounding box orientation because the rectangular shape becomes more stable and recgonizable. On the other hand, Accurate Convex Hull performs comparably with the baseline. While both methods are able to achieve good results (error within 0.2°) when sufficient (e.g. more than 200) points are present, the average difference between the two methods is negligible (lower than 0.1°). In fact, there will be no visually or functionally detectable difference of the two methods when applied to an AD system.

D. *Efficiency and Impact of Collinearity*

While it has been confirmed that using the points from an accurate convex hull does not obviously deteriorate the performance of estimating the bounding box orientation, we then would like to explore the advantage of the method. In this experiment, we only considered test cases with at least 40 points to ensure a relatively accurate output (error

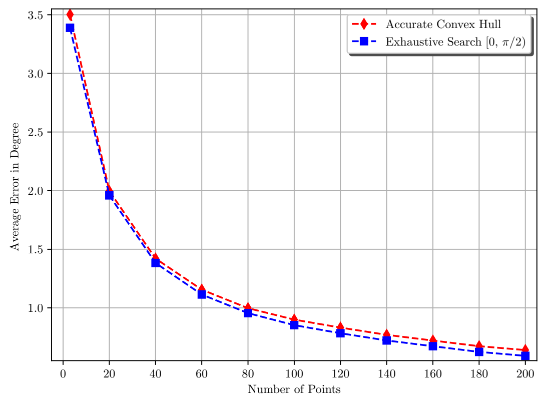
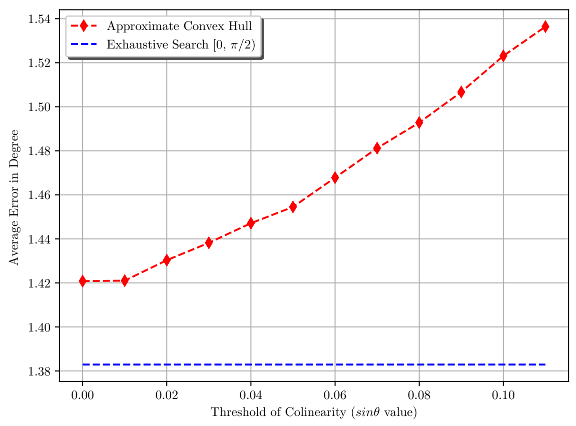
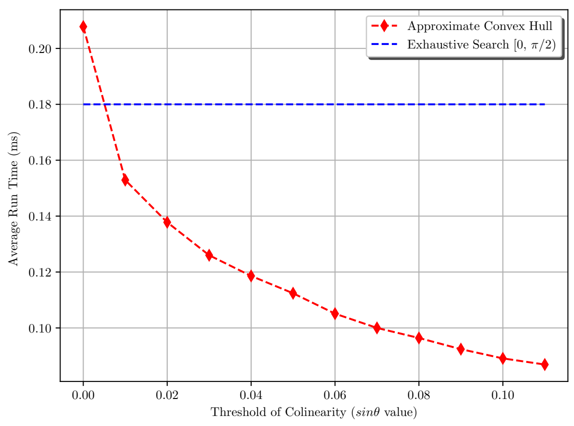


Fig. 2: Performance comparison between Accurate Convex Hull and Exhaustive Search. Both methods provide satisfactory estimation of box orientation when there are sufficient points. The average margin between the two methods is below 0.1°, which is not visually detectable in an AD system.



(a) Performance w.r.t. collinearity



(b) Run time w.r.t. collinearity

Fig. 3: Accuracy and efficiency of using approximate collinearity determination. (a): accuracy w.r.t. collinearity. Error increases when approximate collinearity is applied, however, by only a small amount (0.15°). (b): run time w.r.t. collinearity. A speed-up factor of 2 (compared with the baseline) can be achieved when using 0.10.

within 2°) so as to exclude any bias in evaluation. Figure 3 shows both the average error and run time with respect to the threshold of collinearity determination. The τ value ($\sin\theta$ value in Figure 3 and thereafter) varies from 0 to 0.1, with a step size of 0.01. The horizontal dashed line in both (a) and (b) illustrates the baseline competitor that generates constant values of average error and run time for all the test cases that are studied. The proposed method is named "Approximate Convex Hull" because by using approximate collinearity, some real convex-hull points will be removed. It can be observed in Figure 3(a) that the orientation error will go up when approximate collinearity is used to generate convex-hull points. However, the increasing rate is quite slow, with only 0.1° higher from $\tau = 0$ to $\tau = 0.1$. Even if compared with the baseline method, the error only increases by 0.15°. On the other hand, Figure 3(b) shows that the run time can be significantly reduced, by up to 50% if approximate collinearity is used.

In addition, we further investigated the relationship between efficiency and collinearity, on 3,095 test cases with 1000 points or more. The diamond dashed curve in Figure 4 shows that the run time tends to converge when increasing τ. This can be explained by the squared dashed curve, showing that the convex-hull points cannot be controlled to a small number when there are too many points in a point cloud. We can keep increasing τ to potentially reduce the number of convex-hull points. However, the error will also increase linearly according to Figure 3(a). On the other hand, the time of extracting the convex-hull points ($O(n\log n)$ complexity, see Section III) does not become faster by using a different τ value. As a result, we exploited another technique of downsampling the raw points to a smaller size to improve the algorithm efficiency.

E. Efficiency and Impact of Downsampling

To study the impact of downsampling, we still used the 3,095 test cases with 1000 points or more. For each point cloud, we uniformly downsampled its points to a smaller size ranging from 20 to 100, with a step size of 10. According to Figure 3(a), $\tau = 0.01$ was used in this experiment to generate approximate convex-hull points because it produces almost the same results as using accurate convex-hull points. Figure 5 shows that the proposed method performs comparably with the baseline, with an average difference of 0.04°. By downsampling the raw points to a size of 80 or 100, it works well with an average error lower than 1°.

Finally, Figure 6 shows the run time with respect to the points. By downsampling the points to 100 or fewer, we only need $0.01ms$ to generate the convex-hull points (lower part of Figure 6), usually fewer than 10 if using $\tau = 0.01$ or larger, resulting in $0.03ms$ to estimate the bounding box (upper part of Figure 6). The total run time is $0.04ms$, gaining a speed-up factor of 8 with an error of 0.5°, compared with fitting a bounding box with the raw 1000 points using the baseline method, which takes $0.33ms$ with an error of 0.25°.

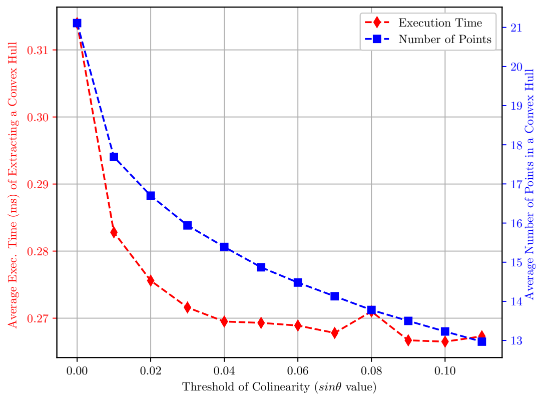


Fig. 4: Run time w.r.t. collinearity on test cases with at least 1000 points. By increasing τ, it is no longer effective to reduce the run time.

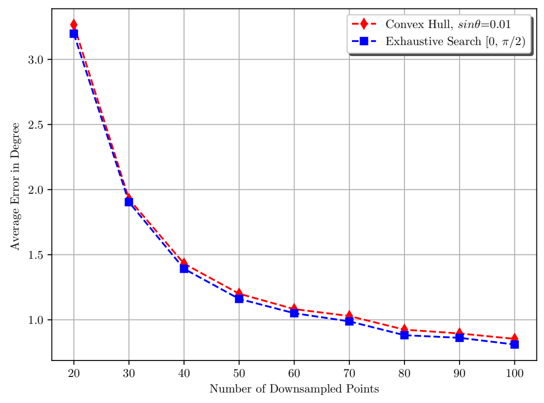


Fig. 5: Comparison of downsampling performance between Approximate Convex Hull ($\tau = 0.01$) and Exhaustive Search, on test cases with 1000 points or more. The average margin between the two methods is below 0.1°. When the number of downsampled points is 80, average errors are within 1°.

V. CONCLUSIONS

We have presented a method of using convex-hull points to estimate 2D bounding box from the bird's-eye view LiDAR points. The method evaluates each candidate angle formed by a pair of points in the convex-hull point set, and selects the orientation that can optimize a given criterion. We proposed different techniques, including approximate collinearity determination and downsampling, to improve the algorithm efficiency. Comprehensive experimental results show that without obviously impacting the performance, the method can reduce the bounding box estimation time by almost one order of magnitude, relative to the naive search-based method that is built upon a fine partition of the angle range. The method can be applied to an autonomous driving system that has high requirement on real-time implementation.

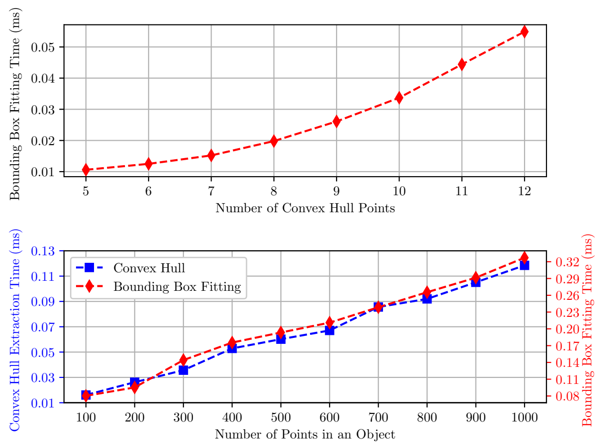


Fig. 6: Run time w.r.t. number of points. Upper: time of fitting a bounding box w.r.t the number of convex-hull points. Lower: time of extracting convex-hull points and fitting a bounding box w.r.t the number of raw object points.

In future, we will investigate generic bounding box estimation method that can be efficiently applied on noisy data, to handle various classes in more cluttered environment.

REFERENCES

[1] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, STD: Sparse-to-Dense 3D Object Detector for Point Cloud, *IEEE International Conference on Computer Vision*, 2019.

[2] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, PointPillars: Fast Encoders for Object Detection from Point Clouds, *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[3] S. Shi, X. Wang, and H. Li, PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud, *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[4] Y. Yan, Y. Mao, and B. Li, SECOND: Sparsely Embedded Convolutional Detection, *Sensors*, 18(10):3337, 2018.

[5] B. Yang, W. Luo, and R. Urtasun, PIXOR: Real-time 3D Object Detection from Point Clouds, *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[6] Y. Zhou, and O Tuzel, VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection, *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[7] M. Simon, S. Milz, K. Amende, and H.-M. Gross, Complex-YOLO: An Euler-Region-Proposal for Real-time 3D Object Detection on Point Clouds, *European Conference on Computer Vision*, 2018.

[8] H. Wang, B. Wang, B. Liu, X. Meng, and G. Yang, Pedestrian recognition and tracking using 3D LiDAR for autonomous vehicle, *Robotics and Autonomous Systems*, vol. 88, February 2017, pp. 71-78.

[9] D. Zermas, I. Izzat, and N. Papanikolopoulos, Fast Segmentation of 3D Point Clouds: A Paradigm on LiDAR Data for Autonomous Vehicle Applications, *IEEE International Conference on Robotics and Automation*, 2017.

[10] M. Himmelsbach, T. Luettel, and H.-J. Wuensche, Real-time Object Classification in 3D Point Clouds Using Point Feature Histograms, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[11] R. A. MacLachlan, and C. Mertz, Tracking of moving objects from a moving vehicle using a scanning laser rangefinder, *IEEE Intelligent Transportation Systems Conference*, 2006.

[12] A. Petrovskaya, and S. Thrun, Model Based Vehicle Tracking for Autonomous Driving in Urban Environments, *Robotics: Science and Systems*, 2008.

[13] X. Shen, S. Pendleton, and M. H. Ang, Efficient L-shape fitting of laser scanner data for vehicle pose estimation, *IEEE Conference on Robotics, Automation and Mechatronics*, 2015.

[14] H. Zhao, Q. Zhang, M. Chiba, R. Shibasaki, J. Cui, and H. Zha, Moving Object Classification using Horizontal Laser Scan Data, *IEEE International Conference on Robotics and Automation*, 2009.

[15] X. Zhang, W. Xu, C. Dong, and J. M. Dolan, Efficient L-shape fitting for vehicle detection using laser scanners, *IEEE Intelligent Vehicles Symposium*, 2017.

[16] D. Kim, K. Jo, M. Lee, and M. Sunwoo, L-shape model switching-based precise motion tracking of moving vehicles using laser scanners, *IEEE Transactions on Intelligent Transportation Systems*, 19(2), 2018.

[17] E. Nezhadarya, Y. Liu, and B. Liu, BoxNet: A Deep Learning Method for 2D Bounding Box Estimation from Bird's-Eye View Point Cloud, *IEEE Intelligent Vehicles Symposium*, 2019.

[18] K. Liu and J. Wang, Fast Dynamic Vehicle Detection in Road Scenarios Based on Pose Estimation with Convex-Hull Model, *Sensors*, 19(14):3136, 2019.

[19] R. A. Jarvis, On the Identification of the Convex Hull of a Finite Set of Points in the Plane, *Information Processing Letters*, 2(1), pp. 18-21, 1973.

[20] R. L. Graham, An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set", *Information Processing Letters*, 1(4), pp. 132-133, 1972.

[21] T. M. Chan, Optimal Output-Sensitive Convex Hull Algorithms in Two and Three Dimensions", *Discrete and Computational Geometry*, 16, pp. 361-368, 1996.

[22] A. Geger, P. Lenz, C. Stiller, and R. Urtasun, Vision meets robotics: The KITTI dataset, *The International Journal of Robotics Research*, 32(11), 2013.