2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

An Orientation Corrected Bounding Box Fit Based on the Convex Hull under Real Time Constraints

Benjamin Naujoks and Hans-Joachim Wuensche

***Abstract*— An important requirement for safe autonomous driving is the perception of dynamic and static objects. In urban scenarios, there exist hundreds of potential obstacles. Therefore, it is crucial to have a fast and accurate fitting method which is a key step for many tracking algorithms. In this paper, we demonstrate an orientation corrected bounding box fit based on the convex hull and a line creation heuristic. Our method is capable of fitting hundreds of objects in less than 10 ms and involves only few tuning parameters. Furthermore, orientation estimated through the dynamics of the object can be used to improve the fitting result. Real-world experiments have proven the robustness and effectiveness of our method.**

I. Introduction

Nowadays, autonomous driving is a popular topic around the world and has initiated a vast amount of research in the last decades [1], [2]. An important requirement for safe navigation of autonomous cars is the perception of dynamic and static objects, which potentially can be obstacles.

The Velodyne HDL-64 LiDAR [3] with its dense three-dimensional point cloud is a popular sensor in the autonomous driving research community [1]. Typically, the dense point cloud is first segmented into ground plane or obstacles and then, clustered into separate point clouds corresponding to objects. These clustered point clouds are usually fitted into bounding boxes to get the position and dimension of the object, which in turn can be associated to a target state for different filtering algorithms such as the Gaussian Mixture Probability Hypothesis Density Filter [4] or the Labeled Multi Bernoulli Filter [5]. We use the previously described LiDAR sensor with a cycle time of 100 ms. Therefore, we consider our algorithms to be real-time if they run within this time period. The computation time of the bounding box fit should be as small as possible, because it is only a small part of the whole estimation chain. Hence, the algorithm of this paper focuses on the runtime.

In this paper, we first create a convex hull of the object's point cloud, obtained with e. g., the method of [6], and then apply Rotating Calipers [7] to get the minimum area rectangle.

For this process, the point cloud is projected in the $x - y$ plane. In most cases, this rectangle does not have the correct orientation as the point cloud covers only a fraction of the whole object. Therefore, we use a heuristic line creation method to get a correct orientation fit. A falsely oriented fit can lead to various problems, e.g, in predicting free space for overtaking maneuvers.

All authors are with the Institute for Autonomous Systems Technology (TAS) of the Universität der Bundeswehr Munich, Neubiberg, Germany. Contact author email: benjamin.naujoks@unibw.de

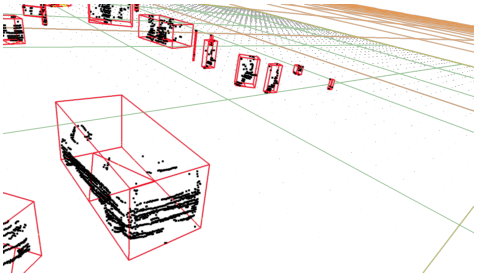


Fig. 1: The resulting three-dimensional bounding boxes fitted on segmented point clouds.

This paper is structured as follows. Section II presents an overview of related research work. Basic definitions used in this paper are described in section III. The algorithm with two alternatives for orientation computation is explained in section IV. In section V, we test our methods in on-road experiments and compare it to PCA [8] and RANSAC [9] based approaches.

II. Related Work

The problem of calculating an Oriented Bounding Box (OBB) can be divided into two separate goals.

The first one is to pursue mathematical optimality, which is the main goal in the geometric community. In the two-dimensional case, this can be done in linear time complexity by Rotating Calipers [7], [10]. For the three-dimensional case, the current best and exact algorithm was published by O'Rourke [11]. Unfortunately, the algorithm has a cubic time complexity with respect to the size of the point set. Therefore, many heuristic approaches have been developed to obtain a practically usable algorithm. Nevertheless, all the algorithms are too slow to fit about 100 OBBs in real-time. For a good comparison, see [10].

The second goal is to account for incomplete data. This occurs in sensor generated point clouds, as they have to deal with occlusions. The most common approach is to do a L-shape fitting [12]. This has the advantage, that it can be applied to various LiDAR sensors. In [13] the L-shape fit is corrected with a search-based optimization. This approach takes almost 3 ms on average per object, which is too slow for tracking multiple objects. Moreover, [14] estimates the orientation with an Extended Kalman Filter based on the symmetrical geometry of the objects and the Rotating Calipers. This approach cannot be used as building block for other processing pipelines as it is not applicable to single-frame methods.

All these approaches have proved their usability in their fields. Nevertheless, our main contributions compared to existing methods are as follows. First, our algorithm is sensor independent and can be used with fused point clouds or point clouds from stereo cameras. Second, we have developed an algorithm which can fit hundreds of objects in real-time and has no need for careful hand-tuning of many parameters. Moreover, our algorithm is deterministic and therefore, can be used for machine learning tasks. Additionally, our approach is robust even for non L-shaped objects as we propose a line search based on the two-dimensional median. In general, it is single-frame based, which is a necessity for the initialization for many tracking algorithms. Furthermore, we propose a tracking based extension to account for sensor uncertainty. Last but not least, the base for our approach is the mathematical optimal minimum area rectangle of Rotating Calipers.

III. Preliminaries

This section describes the basic definitions frequently used in this paper. Let $\boldsymbol{P} = \{\boldsymbol{p}_i\}_{i=1}^{n}$ be a finite set in the two-dimensional plane with $n \in \mathbb{N}$ points $\boldsymbol{p}_i \in \mathbb{R}^2$. Then, the convex hull $CH(\boldsymbol{P})$ of the set is defined as:

$$CH(\boldsymbol{P}) = \left\{ \sum_{i=1}^{n} \alpha_i \boldsymbol{p}_i \mid \alpha_i \in \mathbb{R}_{\geq 0} \wedge \sum_{i=1}^{n} \alpha_i = 1 \right\}, \quad (1)$$

where α_i, $i = 1, \ldots, n$ are weighting factors. Furthermore, $l_{\boldsymbol{p}_i\boldsymbol{p}_j} = \overrightarrow{\boldsymbol{p}_i\boldsymbol{p}_j} \in \mathbb{R}^2$ denotes the line between the two points p_i, p_j. l_x and l_y are the x-/y-component of the line l. The foot point $foot_{l_{\boldsymbol{p}_i\boldsymbol{p}_j}}(\boldsymbol{p})$ of $\boldsymbol{p}$ corresponding to $l_{\boldsymbol{p}_i\boldsymbol{p}_j}$ is calculated through:

$$foot_{l_{\boldsymbol{p}_i\boldsymbol{p}_j}}(\boldsymbol{p}) = \boldsymbol{p}_i - \frac{\langle \boldsymbol{p_i} - \boldsymbol{p}, l_{\boldsymbol{p}_i\boldsymbol{p}_j} \rangle}{\langle l_{\boldsymbol{p}_i\boldsymbol{p}_j}, l_{\boldsymbol{p}_i\boldsymbol{p}_j} \rangle} \cdot l_{\boldsymbol{p}_i\boldsymbol{p}_j}. \quad (2)$$

Consequently, the relative distance $d(\boldsymbol{p}, l_{\boldsymbol{p}_i\boldsymbol{p}_j})$ of $\boldsymbol{p}$ to a line $l_{\boldsymbol{p}_i\boldsymbol{p}_j}$ is defined through the following kind of the Hesse normal form:

$$d(\boldsymbol{p}, l_{\boldsymbol{p}_i\boldsymbol{p}_j}) = \left\langle \tilde{n}_{l_{\boldsymbol{p}_i\boldsymbol{p}_j}}, \boldsymbol{p} - \boldsymbol{p}_i \right\rangle, \quad (3)$$

where $\tilde{n}_{l_{\boldsymbol{p}_i\boldsymbol{p}_j}}$ is the normal vector of $l_{\boldsymbol{p}_i\boldsymbol{p}_j}$. Let $left(l)$ and $right(l)$ denote the set of points which lie on the left hand side, resp. the right hand side of the line. Then, it holds with the definition of the relative distance in Equation 3:

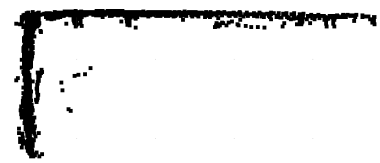
$$left(l) = \left\{ \boldsymbol{p} \in \mathbb{R}^2 \mid d(p, l) > 0 \right\} \quad (4)$$

$$right(l) = \left\{ \boldsymbol{p} \in \mathbb{R}^2 \mid d(p, l) < 0 \right\}. \quad (5)$$

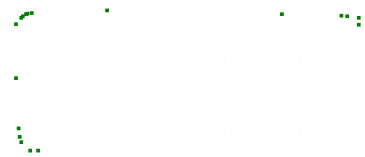
Moreover, the set of the interior points of a rectangle is defined as $\mathcal{I}(R)$, where R is a rectangle defined through its corners. Furthermore, $\mathcal{S}(R)$ is the set of all lines corresponding to the four sides of a rectangle R.

IV. Algorithm

In this section, we explain the algorithm for fitting the bounding box around a segmented point cloud. At first, for computational ease we fit a two-dimensional rectangle by only using the x and y information of the points. Afterwards, we use the z values of the point to get the height of the three-dimensional bounding box.



(a) Segmented point cloud of an object.



(b) Corresponding convex hull of the point cloud.

Fig. 2: Example of a segmented point cloud and its convex hull.

A. The Minimum Area Rectangle

1) Convex Hull: It is shown in [15] that the minimum area enclosing rectangle of a polygon is the same as of its convex hull. Therefore, we want to reduce the overall computation time by using the convex hull of the point cloud. This results in potentially much fewer points to process. The base for the creation of the convex hull $CH(\boldsymbol{P})$ is the Graham-Scan algorithm [16]. Nevertheless, before applying Graham-Scan we filter out the interior points of the rectangle $R_{\max}$ with the corners $\{p_{x_{\min}}, p_{y_{\min}}, p_{x_{\max}}, p_{y_{\max}}\}$, where $p_{x_{\min}}$, $p_{y_{\min}}$ are the points with the minimum x, resp. y values and $p_{x_{\max}}, p_{y_{\max}}$ are the points with the maximum x, resp. y values of the point cloud. Now, we use the following condition to determine the interior points:

$$\boldsymbol{p} \in \mathcal{I}(R_{\max}) \Leftrightarrow (\boldsymbol{p} \in left(l) : \ \forall l \in \mathcal{S}(R_{\max})). \quad (6)$$

With this filtering step the computational load for the convex hull creation is potentially reduced without changing the result of the creation. Figure 2 shows $CH(\boldsymbol{P})$ created from a segmented point cloud corresponding to an object.

2) Rotating Calipers: Rotating Calipers [7] constructs the minimum area rectangle of a convex polygon in linear time. The base for this algorithm is the following theorem:

Theorem 1: The minimum area rectangle enclosing a convex polygon has a side collinear with some edge of the polygon [7].

An enclosing rectangle is a rectangle whose interior contains the interior of the polygon [7], whereas an encasing rectangle is an enclosing rectangle constructed from a pair of parallel lines of support and another pair of parallel lines of support which are perpendicular to the first. Furthermore, a line of support for a simple polygon $\boldsymbol{P}$ is defined as a line L meeting the boundary of $\boldsymbol{P}$, such that $\boldsymbol{P}$ lies entirely on one side of L [7]. Figure 3 shows an example of an encasing rectangle (black) which is built from the parallel lines of support, whereas an enclosing rectangle is illustrated in blue. From Theorem 1, we can conclude that the minimum enclosing

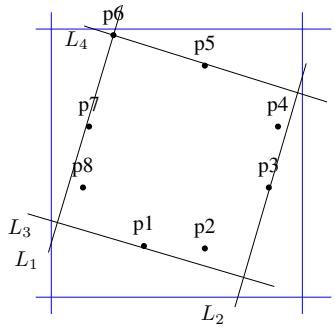


Fig. 3: Possible convex hull points, the encasing rectangle (black) built from parallel lines of support namely L_1, L_2, L_3 as well as L_4 and the enclosing rectangle (blue).

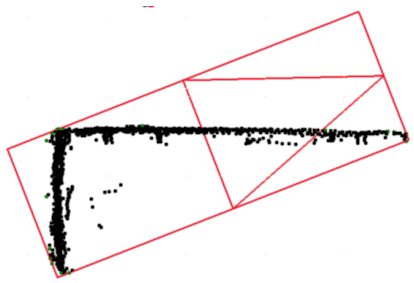


Fig. 4: The minimum area rectangle of the segmented point cloud.

rectangle is the encasing rectangle anchored on an edge of the polygon with the minimum area [7]. Therefore, we only need to find the minimum encasing rectangle which implies a linear complexity with respect to the number of convex hull points. For the whole algorithm, see for example [7]. The result of applying Rotating Calipers to the convex hull of a segmented point cloud is shown in Figure 4. Clearly, the rectangle does not have the correct orientation and therefore, further steps have to be applied.

B. Orientation Corrected Bounding Box

Despite the mathematical optimality of the minimum area rectangle, in most cases it is not the desired fit for segmented point clouds, as the point cloud hardly covers the whole object. Therefore, additional refinement steps are necessary, especially for the orientation of the fit. In the following we describe the essential steps to refine the fit.

1) Method 1: Calculating the Angle: Here, we want to calculate the orientation difference through a characteristic line. There are several techniques for line creation, such as incremental or RANSAC based algorithms [17]. We use a three point based heuristic approach. The main advantages are less runtime and the deterministic computation. The base for $(\boldsymbol{p}_l, \boldsymbol{p}_h)$, the lower and higher point, are the diameter points of the constructed convex hull, which are in fact the diameter points of the two dimensional point cloud. This can be done in linear time with respect to the number of convex hull points, by a variant of rotating calipers described in [18]. Figure 5 shows the calculated points colored in yellow and

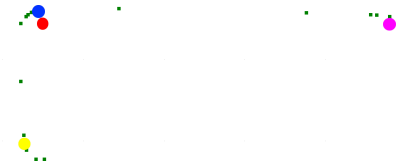


Fig. 5: The convex hull with the lower extreme point (yellow), the higher extreme point (magenta), the third point (blue) and the two-dimensional median (red).

magenta. The third point is calculated as follows:

$$\boldsymbol{p}_3 = \underset{\boldsymbol{p} \in CH(\boldsymbol{P})}{\operatorname{argmax}} \left(d(l_{\boldsymbol{p}_l, \boldsymbol{p}_h}, \boldsymbol{p}) - \lambda \cdot \| foot_{l_{\boldsymbol{p}_l, \boldsymbol{p}_h}}(\boldsymbol{p}) - \boldsymbol{p}_e \|_2 \right), \tag{7}$$

where $\| foot_{l_{\boldsymbol{p}_l, \boldsymbol{p}_h}} - \boldsymbol{p}_e \|_2 = \min(\| foot_{l_{\boldsymbol{p}_l, \boldsymbol{p}_h}} - \boldsymbol{p}_l \|_2, \| foot_{l_{\boldsymbol{p}_l, \boldsymbol{p}_h}} - \boldsymbol{p}_h \|_2)$ is the minimum distance of the foot point of $\boldsymbol{p}$ on the line and one of the diameter points. Last but not least $\lambda \in \mathbb{R}$ is a tuning parameter. A calculated $\boldsymbol{p}_3$ is illustrated as blue point in Figure 5. Now, there are three possible lines for the angle calculation. Namely, $\overrightarrow{p_h p_l}$, $\overrightarrow{p_h p_3}$ and $\overrightarrow{p_l p_3}$. The main criteria for choosing the right line is the two dimensional pseudo-median of all points. Let $n \in \mathbb{N}$ be the size of the point cloud and $\boldsymbol{w} = \{w_k\}_{k=1}^n$ the weights associated to the points. Furthermore, $\circ$ denotes the Hadamard product. Then, the algorithm for determining the pseudo-median is described in Algorithm 1. The two-dimensional $median(\boldsymbol{w} \circ \boldsymbol{p})$ is the

Algorithm 1 Iterative Median Algorithm

```
for i = 0 : #Iterations do
    if i == 0 then
        w_k = 1, k = 1, ..., n
    else
        w_k = ||p_median − p_k||, k = 1, ..., n
    end if
    p_median = median(w ∘ p),
end for
```

concatenation of the median of all x-values and the median of all y-values. p_{median} of the example point cloud is shown in Figure 5 denoted by the color red. Clearly, the median is more likely to lie near the majority of all points, compared to their mean. Moreover, in the most cases two iterations are enough to get a robust median near the majority of line points. This is the main motivation for using the iterative median. After the median computation the correct line $l \in \mathbb{R}^2$ is chosen according to the following criteria:

$$l = \underset{\hat{l} \in \{\overrightarrow{p_h p_l}, \overrightarrow{p_h p_3}, \overrightarrow{p_l p_3}\}}{\operatorname{argmin}} \frac{\| \hat{l} - p_{median} \|_2}{\| \hat{l} \|_2}. \tag{8}$$

Consecutively, we calculate the orientation difference $\Delta\alpha \in \mathbb{R}$. Let $\beta \in \mathbb{R}$ be the orientation of the corresponding side of the minimum area rectangle of IV-A. Then, it follows:

$$\Delta\alpha = \beta - \arctan(l_y, l_x), \tag{9}$$

where l_x, l_y are the $x-$, respectively $y-$ components of the line.

2) Method 2: Estimated Angle: In the last part, we calculated the orientation difference in a heuristically manner. Another approach is to estimate the true orientation based on a Kalman Filter [19] depending on the dynamics of the object. There are several dynamic models to estimate the orientation. For tracking a car, the Ackermann process model [20] is best suited, whereas the Augmented Coordinate Turn (ACT) [20] generates better results for pedestrians as they can rotate around their own axis. For non-linear models the Extended Kalman Filter [21] or the Unscented Kalman Filter [22] are advisable. If we have the estimate of the true orientation $\alpha_{estimate}$, calculating the difference is straightforward with: $\Delta\alpha = \beta - \alpha_{estimate}$, where β is defined as in Equation 9. It is important to note that this approach is only preferable for moving objects as the orientation is estimated through the dynamic model.

3) Rotation of the Rectangle: The next step is to rotate the corners $E_{bb} = \{e_1, e_2, e_3, e_4\}$, $e_i \in \mathbb{R}^2$ of the previously computed bounding box with the orientation difference $\Delta\alpha$. Therefore, we build the two-dimensional rotation matrix $R(\phi) \in \mathbb{R}^{2\times 2}$ with $\phi \in \mathbb{R}$.

$$R(\phi) = \begin{pmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{pmatrix}. \tag{10}$$

Additionally, we have to account for the rotation anchor point $p_{anchor} \in \{p_l, p_h\}$, as we do not want to rotate around the bounding box center. p_{anchor} equals p_l if p_3 is closer to p_l and p_h otherwise. Hence, the steps for the rotation of the bounding box corners are as described in Algorithm 2.

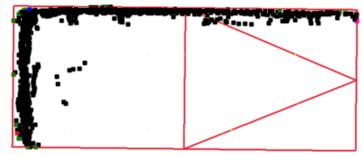
Algorithm 2 Rotate Corners

for $e \in E_{bb}$ **do**
  $e -= p_{anchor}$
  $e = e \cdot R(-\Delta\alpha)$
  $e += p_{anchor}$
end for

4) Dimension Correction: After the rotation of E_{bb} we have to correct the rectangle dimensions, as when using the modified orientation the rectangle is either not tightly fitted or too narrow. This can be seen in Figure 6a. There are two cases for every bounding box side. In the first case, the side is too narrow. This means that there exist convex hull points at the right hand side of the line. In the second case, the side is not tight enough. Define $CH_{right(l)}(\boldsymbol{P}) = \{\boldsymbol{p} \in CH(\boldsymbol{P}) \mid \boldsymbol{p} \in right(l)\}$. Then, we choose the desired point $\boldsymbol{p}^* \in CH(\boldsymbol{P})\ \forall l \in \mathcal{S}(R)$, with R is the fitted rectangle with the corners $\{e_1, e_2, e_3, e_4\}$, through:

$$\boldsymbol{p}^* = \begin{cases} \max\limits_{\boldsymbol{p} \in CH_{right(l)}(\boldsymbol{P})} d(\boldsymbol{p}, l) & \exists \boldsymbol{p} \in CH_{right(l)}(\boldsymbol{P}) \\ \min\limits_{\boldsymbol{p} \in CH(\boldsymbol{P})} d(\boldsymbol{p}, l) & \text{otherwise} \end{cases}. \tag{11}$$

(a) The orientation corrected bounding box. Clearly, the dimensions have to be corrected.



(b) The dimension corrected rectangle. Now, the rectangle is a tight fit with correct orientation.

Fig. 6: Before (left) and after (right) the dimension correction.

Afterwards, we correct the position of the corners according to:

$$e_1 += \boldsymbol{p}^*_{12} - foot_{l_{e_1e_2}}(\boldsymbol{p}^*_{12}) \tag{12}$$
$$e_2 += \boldsymbol{p}^*_{12} - foot_{l_{e_1e_2}}(\boldsymbol{p}^*_{12}) \tag{13}$$
$$e_2 += \boldsymbol{p}^*_{23} - foot_{l_{e_2e_3}}(\boldsymbol{p}^*_{23}) \tag{14}$$
$$e_3 += \boldsymbol{p}^*_{23} - foot_{l_{e_2e_3}}(\boldsymbol{p}^*_{23}) \tag{15}$$
$$e_3 += \boldsymbol{p}^*_{34} - foot_{l_{e_3e_4}}(\boldsymbol{p}^*_{34}) \tag{16}$$
$$e_4 += \boldsymbol{p}^*_{34} - foot_{l_{e_3e_4}}(\boldsymbol{p}^*_{34}) \tag{17}$$
$$e_4 += \boldsymbol{p}^*_{41} - foot_{l_{e_4e_1}}(\boldsymbol{p}^*_{41}) \tag{18}$$
$$e_1 += \boldsymbol{p}^*_{41} - foot_{l_{e_4e_1}}(\boldsymbol{p}^*_{41}). \tag{19}$$

5) 3D Bounding Box: All the previous sections are two-dimensional approaches. To get the three-dimensional bounding box, we calculate the height of the object $h \in \mathbb{R}$ with:

$$h = z_{\max} - z_{\min}, \tag{20}$$

where $z_{\max}$ and $z_{\min}$ are the maximum and minimum value of the z-dimension of the original three-dimensional point cloud. This can be done in linear time with respect to the size of the segmented point cloud.

The resulting bounding box is shown in Figure 1. Of course, this approach does not handle outliers e.g points of a hanging tree branch above the car. There exist several approaches to overcome this problem. In example, [12] does a weighted least-squares fit to reject such outliers at feature creation step.

C. Asymptotic Runtime

The asymptotic runtime is an important indicator for the computational load of an algorithm. Clearly, the overall runtime is dominated by the sorting step in the convex hull as well as median computation, which is logarithmic with respect to the size of the point cloud. The other steps are mainly linear with respect to the size of the convex hull.

V. Results

This section compares our method with a simple PCA based approach [8] and a RANSAC based method [9]. The following tests were made in a real-world urban environment. Example video footage can be found online[1]. We used a standard office computer with an Intel(R) Core i7 CPU. The algorithms were implemented in our institutes C++ Framework. Furthermore, there was no parallelization enabled. For our experiments, we used our institutes autonomous car MuCAR-3 [23] with roof-mounted Velodyne HDL64-S3. To obtain the ground truth data for the fitting result comparison, we installed an INS-sensor into our ground truth object and recorded the position, orientation as well as the dimension for every time step. Next, we segmented the point cloud by erasing all points which don't lie inside the ground truth bounding box. Then, we fitted the segmented point cloud with the different methods. Here, we set the parameter λ of Equation 7 to 0.01. For the estimated angle of section IV-B.2, we used the same tracking algorithm as described in [23].

A. Runtime Comparison

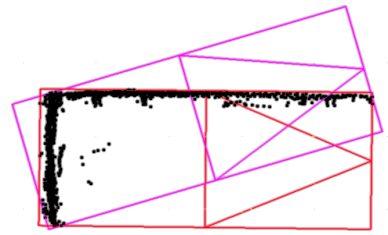
Table I compares the mean running time of the fitting methods. Here, the same point cloud of 91 objects of different sizes up to $15\,\mathrm{m}$ were fitted 10000 times to get a reliable mean run-time for every method. The corresponding point cloud size of one object varies between 2 and 1884 with a mean size of 83. PCA is the fastest, but our method is approximately one magnitude faster than the RANSAC method of [9]. Hence, in terms of computational load, the PCA based approach is the best choice, while our approach is capable of fitting almost 100 objects in less than three milliseconds. Additionally, the runtime can be further reduced by applying parallelization.

TABLE I: Comparison of the runtime average for 91 objects over 10000 iterations.

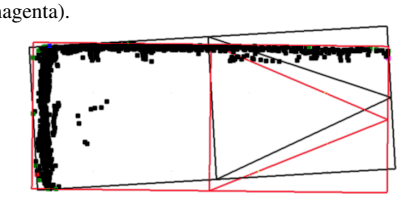
Method	Mean time in ms
Our method	2.693
RANSAC based	13.258
PCA based	0.284

B. Fitting Results

In contrast to the runtime, the fitting result of PCA is inferior in the most cases, especially for L-shapes. This can be seen in Figure 7a. Moreover, Figure 7b shows a comparison of the fitting results of our method and the RANSAC approach. Whereas, Figure 8 shows that, in contrast to other heuristics [13], our approach works well even for non-L-shaped point clouds. As this are just examples we show a histogram of orientation errors for a test run lasting 233 seconds with 2331 frames, where the orientation error is defined as the difference between the ground truth orientation and the fitted orientation. The test run included overtaking and turning maneuvers as well as lane changes to generate challenging test data.



(a) Comparison of our method (red) and PCA based approach (magenta).



(b) Comparison of our method (red) and RANSAC based approach (black).

Fig. 7: The evaluated methods applied on a L-shaped point cloud.

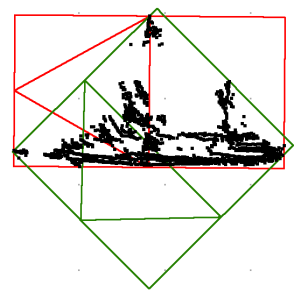


Fig. 8: Our method (red) against the minimum area rectangle (green) for a non L-shaped point cloud.

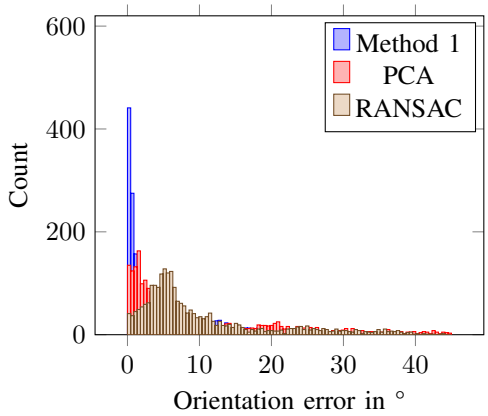
Table II shows, that our methods are superior to the PCA and RANSAC based method. Method 2 has a mean absolute error of $2.78°$ which is comparable to [13], while being significantly faster. Moreover, PCA generates slightly better results than RANSAC through the dense point cloud of the ground truth object. From this follows, that the segmented point cloud has been rarely in a L-shape, because in this case PCA would create worse results as can be seen in Figure 7a.

TABLE II: Comparison of the mean orientation errors with ground truth object.

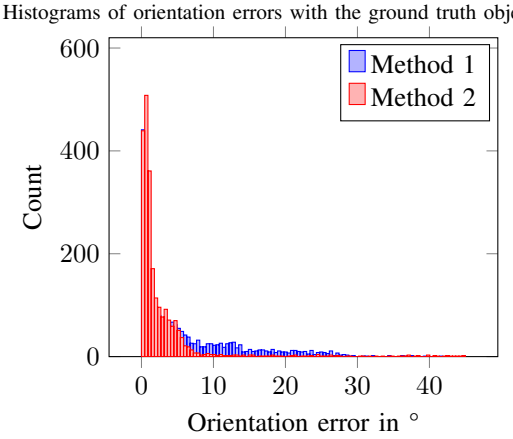
Method	Error (2331 steps)
Method 1	5.89°
Method 2	2.78°
PCA	9.91°
RANSAC	10.21°

Figure 9a displays the same. Method 1 is densely distributed around the smaller error values, whereas RANSAC and PCA have a wider distribution. Moreover, the error distribution of method 2 is more densely packed around

[1] http://www.mucar3.de/iv2018-ocbbf



(a) Histograms of orientation errors with the ground truth object.



(b) Histograms of orientation errors with the ground truth object 1 and estimated angle.

Fig. 9: Comparison of our two methods, PCA and RANSAC with respect to the orientation errors.

the small error values as with method 1, as can be seen in Figure 9b. Naturally, this modification is only helpful if there are segmented point clouds in consecutive time steps. Therefore, Method 1 was used for the initialization of the estimated orientation. Furthermore, the fitted orientation is highly depending on the tracking algorithm.

VI. Conclusion

In this paper, we proposed a method to fit a bounding box around incomplete segments of point clouds with correct orientation. The method is easy to implement and only has few tuning parameters. Furthermore, it is straightforward to parallelize over the number of objects. Our approach is capable of fitting almost 100 objects in no more than 3 ms without enabling any parallelization. Additionally, our method showed effectiveness even for non L-shaped objects. Moreover, on-road experiments have shown the superiority of our method compared to RANSAC and PCA based approaches as well as competitive results compared to [13], [14] with a mean absolute error of 2.78° using method 2. At these experiments, thousands of different geometric shapes resulting from the views of different maneuvers have been fitted. Nevertheless, even when using heuristic method 1 the average orientation error remains in the single-digits. Additionally, we estimated the orientation through the dynamics of the object. This extension has further reduced the average error.

In our future work, we will research the runtime with different parallelization methods as well as investigate various outlier rejection schemes for the convex hull creation and height determination.

References

[1] S. Thrun, M. Montemerlo *et al.*, "Stanley: The Robot that Won the DARPA Grand Challenge," *J. Field Robotics*, vol. 23, no. 9, 2006.

[2] A. Teichman, J. Levinson, and S. Thrun, "Towards 3D Object Recognition via Classification of Arbitrary Object Tracks," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2011.

[3] Velodyne Lidar, Inc., "High Definition Lidar HDL-64E S2 Specifications." [Online]. Available: http://velodynelidar.com/lidar/hdlproducts/hdl64e.aspx

[4] B.-N. Vo and W.-K. Ma, "The Gaussian Mixture Probability Hypothesis Density Filter," *IEEE Trans. Signal Process.*, vol. 54, no. 11, 2006.

[5] S. Reuter, B.-T. Vo, B.-N. Vo, and K. Dietmayer, "The Labeled Multi-Bernoulli Filter," *IEEE Trans. Signal Process.*, vol. 62, no. 12, 2014.

[6] P. Burger and H.-J. Wuensche, "Fast Multi-pass 3D Point Segmentation Based on a Structured Mesh Graph for Ground Vehicles," in *Proc. IEEE Intelligent Vehicles Symp. (IV)*, Jun. 2018.

[7] D. S. Arnon and J. P. Gieselmann, "A Linear Time Algorithm for the Minimum Area Rectangle Enclosing a Convex Polygon," Purdue University, Department of Computer Science, Tech. Rep., 1983.

[8] H. Zhao, Q. Zhang *et al.*, "Moving Object Classification using Horizontal Laser Scan Data," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2009.

[9] M. Himmelsbach and H.-J. Wuensche, "Tracking and Classification of Arbitrary Objects with Bottom-Up/Top-Down Detection," in *Proc. IEEE Intelligent Vehicles Symp. (IV)*, 2012.

[10] C.-T. Chang, B. Gorissen, and S. Melchior, "Fast Oriented Bounding Box Optimization on the Rotation Group SO(3,$\mathbb{R}$)," *ACM Trans. Graph.*, vol. 30, no. 5, 2011.

[11] J. O'Rourke, "Finding Minimal Enclosing Boxes," *International Journal of Parallel Programming*, vol. 14, no. 3, 1985.

[12] R. MacLachlan and C. Mertz, "Tracking of Moving Objects from a Moving Vehicle using a Scanning Laser Rangefinder," in *Proc. IEEE Intelligent Transportation Syst. Conf. (ITSC)*, 2006.

[13] X. Zhang, W. Xu, C. Dong, and J. M. Dolan, "Efficient L-Shape Fitting for Vehicle Detection Using Laser Scanners," in *Proc. IEEE Intelligent Vehicles Symp. (IV)*, 2017.

[14] P. Kmiotek and Y. Ruichek, "Representing and Tracking of Dynamics Objects using Oriented Bounding Box and Extended Kalman Filter," in *Proc. IEEE Intelligent Transportation Syst. Conf. (ITSC)*, 2008.

[15] H. Freeman and R. Shapira, "Determining the Minimum-area Encasing Rectangle for an Arbitrary Closed Curve," *Commun. ACM*, vol. 18, no. 7, Jul. 1975.

[16] R. L. Graham, "An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set," *Inf. Process. Lett.*, vol. 1, no. 4, 1972.

[17] V. Nguyen, S. Gächter *et al.*, "A Comparison of Line Extraction Algorithms using 2D Range Data for Indoor Mobile Robotics," *Autonomous Robots*, vol. 23, no. 2, Aug. 2007.

[18] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*. New York, NY, USA: Springer-Verlag New York, Inc., 1985.

[19] R. E. Kalman, "New Approach to Linear Filtering And Prediction Problems," *ASME Journal of Basic Engineering*, 1960.

[20] Y. Bar-Shalom, T. Kirubarajan, and X.-R. Li, *Estimation with Applications to Tracking and Navigation*. New York, NY, USA: John Wiley & Sons, Inc., 2002.

[21] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*, ser. Mathematics in Science and Engineering. Acad. Press, 1970, no. 64.

[22] S. J. Julier and J. K. Uhlmann, "Unscented Filtering and Nonlinear Estimation," in *Proc. IEEE*, 2004.

[23] F. Ebert, D. Fassbender, B. Naujoks, and H.-J. Wuensche, "Robust Long-Range Teach-and-Repeat in Non-Urban Environments," in *Proc. IEEE Intelligent Transportation Syst. Conf. (ITSC)*, Yokohama, Japan, 2017.