Joint Detection and Tracking for Automatic Annotation of 3D Data

Ruihao Zhang   Lingrui Zhang   Shuheng Zhang

June 2022

1 Introduction

With the increasing popularity of autonomous driving, we need a lot of accurately labelled data to train our model. In recent years, with the progress of 3D deep learning and strong application requirements, 3D object detection and tracking technology has developed rapidly. One important direction in this field is to let the machine "automatically label" data, to save the cost of manual labeling.

Currently, most 3D perception research has been focusing on real-time use cases and only considers sensor input from the current frame or a few history frames. Those models are sub-optimal for automatic labeling where the best perception quality is needed. However, there are several problems troubling scholars. When target object is far away from detector or it is sheltered by other objects, it may vanish in some frames. In addition, the jitter of boxes also reduce accuracy of objects' detection and tracking.

In this project, we are going to focus on improving accuracy of object detection with assistance of object tracking using multi-frame input based on existing detection and tracking methods. And implement 3D objects tracking with different frames to achieve combination of track and detection. Among above problems, We fix size of boxes to alleviate jitter problem and design a method to solve occlusion problem.

2 Related Work

At present, some well-known commercial companies have proposed some automatic annotation methods for 3D point cloud. Most of them use off-the-shelf object detection [4] and tracking models [5] to generate initial results which will be further optimized. For example, Waymo divides objects into dynamic and static ones for labeling, and fixes bounding boxes of the same static objects [2]. And Uber fixes the size of each object first and then optimizes the trajectory of each object [6].

Both methods above get excellent performance, however, they do not handle an important situation where some frames in a trajectory may be missed due to occlusion or distance. We will combine the advantages of these two methods and try to complement lost frames, meanwhile, we will attempt new methods to improve accuracy of object detection with tracking information.

3 Methodology

We construct a complete system to deal with the problem. In our system (Figure 1), there are two stages, the first stage is application of off-the-shelf models and the second stage is our proposed method.

In the first stage, we adapt PVRCNN [4] to get initial object detection labels and then leverage AB3DMOT [5] to finish object tracking task. What should be attentioned is that we filter those labels with low confidence in order to fit our proposed method better. If the threshold of confidence is too high, we cannot get enough labels to predict bounding boxes well in the following steps while if it is too low, there will be mountains of negative samples which hinders our following work as well. However, threshold of confidence is hard to nail down and we take several experiments on it and finally choose the best one. In fact, apart from models we use in this paper, you can also use other models to complete the

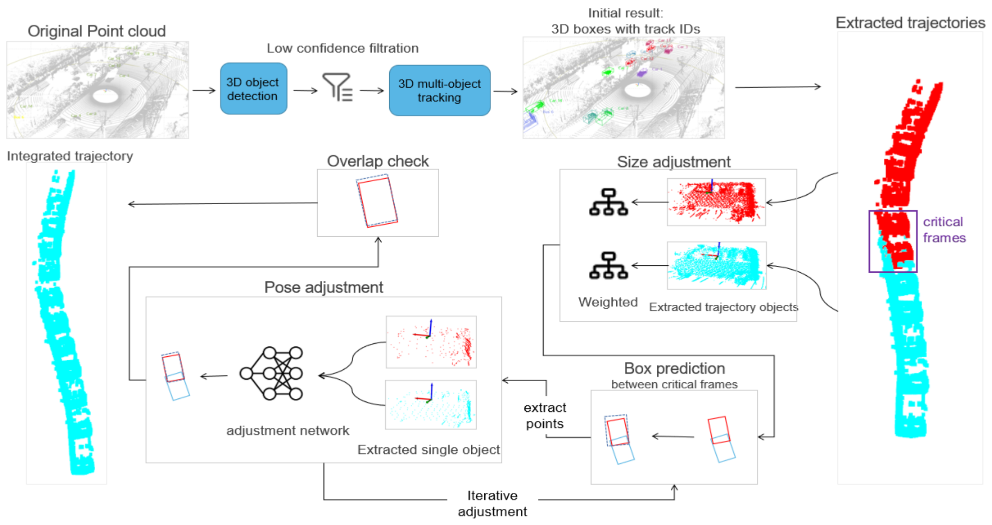


Figure 1: System structure

first stage.

The second stage involves several steps, size adjustment, trajectory complement. If a certain trajectory is split into two trajectories with different id, the second stage can restore the it. Firstly, considering a rigid object has fixed dimensions, we will fix those of all objects in a certain trajectory. To implement it, we will extract all points and corresponding bounding boxes, then use our algorithm to obtain the final dimensions. The second part is trajectory complement. We define frames where the trajectory is divided as critical frames as right side of Figure1 shows. Box prediction algorithm will be applied on critical frames so that we can obtain our network 's inputs which are points in bounding boxes predicted. With extracted points from adjacent frames, it comes to pose adjustment. We construct a innovative network to refine predicted bounding boxes. To ensure complement all frames, box prediction and pose adjustment will be iteratively executed until our network finds there is no expected object in predicted box. Eventually, checking overlap between current trajectory and others, certain two trajectories may merge together like left side of Figure1 shows.

3.1 Low Confidence Filtering

Because of deficiency of current detection models, there will be false positive and false negative labels after applying detection models on a scene which brings great error to our subsequent experiments.

We measured the performance at different confidence levels through experiments. As shown in Experiment 1, confidence is used to reduce false recognition. However, due to the low confidence caused by the occlusion relationship, some objects will sometimes exist. In subsequent methods, these falsely deleted tags are added back to the original trajectory by trajectory complement frame.

3.2 Size Adjustment

To make our detection more accurate, we need to fix the size of each object label. We experimented with many methods as following.

BEV + CNN means that the height information of bounding box is ignored, and then the grid is carried out in 2D plane. Our setting is that each grid is a square with a side length of 5cm, with a total of 128×128 grids. We extracted point clouds from all bounding boxes of a trajectory and aligned them centrally. Thus, we have a 128×128 single-channel image, which we feed into the convolutional neural network. The output is the length and width of the object. As the centers of many frame point clouds need to be overlapped, slight deviations in the positions of some bounding boxes may have a great impact on the results. The unsatisfactory results obtained in our experiment may also be caused by this reason.

In point cloud detection, the object we detect is affected by angle and distance. It not encourage to directly obtain the accurate length and width of the object, but only a compressed or stretched length and width in the current situation. But this has a certain impact on the accuracy of our data final label.

For the best angle method, we will frame-by-frame detect objections with the same ID by processing the tracked results. According to the characteristics of the detected point cloud coordinates, we use important angels α, θ, and rotation along y R_y to calculate its best height and width as Figure 2 shows.

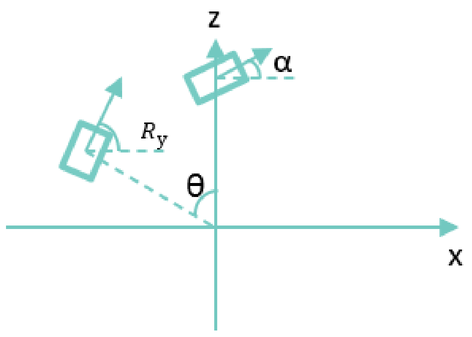


Figure 2: Demo of important angels

We have equation $\theta = R_y - \alpha$. We map θ and α to the sin function, and take their absolute values. As shown in Figure 3, we get the most possible length (left figure) and width (right figure) in the limit case. Under the best situation, α and θ have following equations for length and width (k is odd, t is even):

$$\textbf{Length} \begin{cases} \alpha = t\pi, \ \theta = t\pi \\ \alpha = k\frac{\pi}{2}, \ \theta = k\frac{\pi}{2} \end{cases} \tag{1}$$

$$\textbf{Width} \begin{cases} \alpha = k\frac{\pi}{2}, \ \theta = t\pi \\ \alpha = t\pi, \ \theta = k\frac{\pi}{2} \end{cases} \tag{2}$$

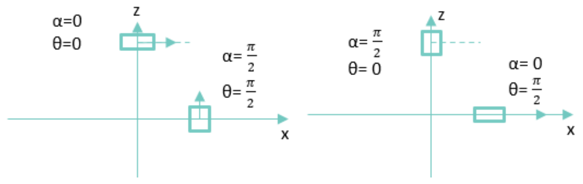


Figure 3: The most suitable length and width

According to the value of the sin function of α and θ, we can iterative obtain the most possible length and width. and replace the old label.

For the best angle + position adjustment method, we need above method and select a corner where the point cloud is denser, and fit it with the label corner of the exact size we get like Figure 4. It can ensures that the center of the new label we get is the center of the actual object.[6] It can also reduce the error caused by recalculating the center point.However, it cannot process the state of object occlusion for we cannot get a well point-intensive areas.

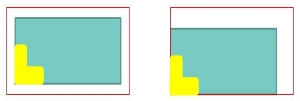


Figure 4: Adjust the location of the label

There are also two very simple methods that have unexpected results. First of all, the average of all bounding box sizes of a trajectory is directly calculated, which can achieve a result similar to the baseline. In addition, we find that weighted averages work better. The size deviation in the detection stage is mainly due to the fact that objects are partially blocked, resulting in small bounding box. If we assign larger weights to larger boxes, we will get better results than the average.

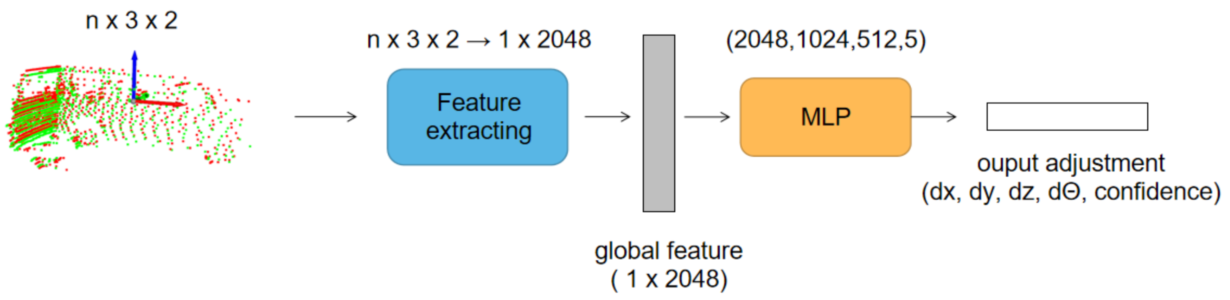


Figure 5: Proposed network

3.3 Trajectory Complement

After the object tracking is finished, we need to check whether each track has premature ending or the same track is represented by multiple IDs because of occlusion. Here are our main methods:

3.3.1 Bounding Box Prediction

First step is to predict a bounding box based on previous frames in the same trajectory. We will predict one next frame for each trajectory and then check whether there is a object in the predicted position. If a object is detected, the bounding box will be integrated into current trajectory and continue predicting next box otherwise we abandon the box and cease current trajectory's process of prediction.

To predict position of bounding box, we use cubic exponential smoothing. Given several previous frames' positions, we can predict the approximate position of the next frame through a smooth curve. Setting smooth coefficient α, and we get the calculation formula

$$s_t = \alpha x_t + (1-\alpha)s_{t-1}$$

with t = 1,2,3. And get the expected calculation parameters

$$a = 3s_1 - 3s_2 + s_3$$

$$b = \frac{\alpha}{2(1-\alpha)^2}[(6-5\alpha)s_1 + 2(5-4\alpha)s_2 + (4-3\alpha)s_3]$$

$$c = \frac{\alpha^2}{2(1-\alpha)^2}[s_1 - 2s_2 + s_3]$$

Then we can use the expected calculation parameters to predict the next position of bounding box by formula

$$prediction = a + b + c$$

.

3.3.2 Pose Adjustment

After predicting bounding box, we apply network to judge whether there is excepted object and then adjust its pose if there is. To judge existence of object, we use PointNet [3] to extract features of point clouds and then apply full connection layer to decode features and finally get possibility of existence. Adjustment involves four variables $dx, dy, dz, d\theta$ which stand for translation in x, y, z and rotation angel $d\theta$. According to our observation, the number of points in box and orientation of box between two adjacent frames are highly related. Therefore, we adapt structure like SiameseNet [1] to combine features of adjacent frames.

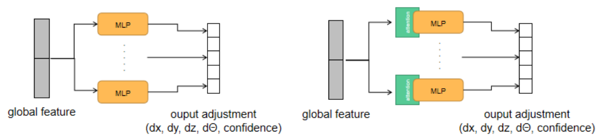


Figure 6: Different headers

Then, we try several strategies to decode such features: single-header method, multiple-headers

Confidence Threshold	TP	FP	FN	Accuracy	Precision	Recall	f1
0.6	28474	16044	11500	0.7136	0.6396	0.7123	0.674
0.7	27390	12647	12724	0.7603	0.6841	0.6828	0.6834
0.8	25158	9039	15163	0.8169	0.7357	0.6239	0.6752
0.9	20788	6065	19680	0.8697	0.7741	0.5136	0.6176

Table 1: Results in different confidence thresholds

method (left figure in Figure 6), multiple-headers-with-attention method (right figure in Figure 6). Single-header approach only takes one header to decode features and finally gets 5 output $dx, dy, dz, d\theta, p$, where p is possibility of existence of object. Multiple-headers method has 5 header for each output and multiple-headers-with-attention method adds attention layers for each headers based on the former in order to focus on important parameters. According to our experiments, we find single-header method has the best performance so we take this network as our final choice. Network for pose adjustment is as Figure 5.

3.3.3 In-trajectory Complement

Some bounding boxes are removed due to low confidence, but actually they are part of the trajectory, which are mistakenly deleted by us. Our tracker basically handles this problem, so some frames are missing in a track. We will select the longest consecutive frames in this trajectory to predict the missing parts. Specific operations are carried out according to the "Bounding Box Prediction" and "Pose Adjustment" mentioned above.

3.3.4 Trajectory Merging

Since trackers are not 100% reliable, especially if parts of the track are missing, the same track can be mistaken for different tracks. After completing each trajectory as continuous in the previous step, we will continuously predict to both ends of the trajectory to judge whether there are objects predicted (Figure 7). If there is an object, adjust its position as before. At this point, we also need to determine whether the object at this position is already in another trajectory. If the object is already in another track, we will merge the two tracks into the same ID. The judgment criterion here is that the distance between the predicted box and the existing boxes in other trajectories of the same frame is less than 0.3m, and the angle difference is less than 0.3 radian, and they correspond to the same object type, so the two boxes are considered to correspond to the same object.

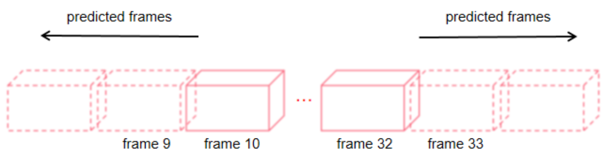


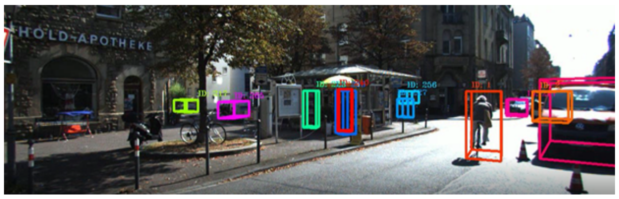
Figure 7: Diagram of prediction method

If boxes are successfully added at one end or both ends of the trajectory, then we believe that the trajectory is still likely to expand to both sides, so we will continue to predict until no new boxes can be added to both sides, and the trajectory is considered complete.

4 Experiments

4.1 Threshold Selection

After detecting a scene, we further refine the trajectory by selecting the confidence, which can reduce the number of false positives. However, it is very important to determine the appropriate confidence threshold. If the threshold is too high, a large number of correct results are deleted ; if the threshold is too low, it is difficult to achieve the desired results. Therefore, we conducted experiments to explore the


(a) Some error cases

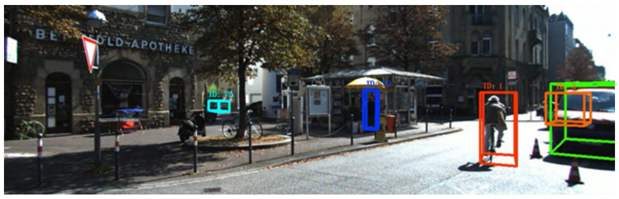

(b) After removing results with low confidence

Figure 8: Low-confidence Removal

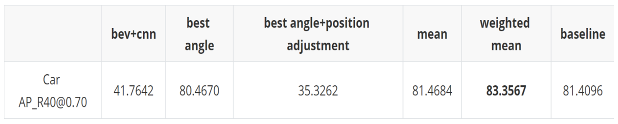
impact of different confidence thresholds on results, as shown in the Table 1.

As shown in the table above, if the confidence is low, there will be a large number of false positives, and in our method, false negatives can be made up in the subsequent method. So, we should choose between 0.8 and 0.9. 0.8 compared with 0.7, the decrease of false positives and increase of false negatives are roughly the same. But 0.9 compared with 0.8, the decrease of false positives is much less than the increase of false negatives, and the number of false negatives is too large. This may lead to too many bounding boxes missing, and the trajectory information is not enough for us to complete them. So, we decide to use 0.8 as the threshold to remove low confidence.The result is shown in Figure 8. We can clearly find that the number of false detection has decreased a lot.

4.2 Size Adjustment

The bounding box size is initially determined by detection, so only single frame information is considered. We hope to use multi-frame information to determine the size of rigid objects.

We propose a variety of possible approaches we mentioned in the methodology, as shown in Figure 9.



	bev+cnn	best angle	best angle+position adjustment	mean	weighted mean	baseline
Car AP_R40@0.70	41.7642	80.4670	35.3262	81.4684	**83.3567**	81.4096

Figure 9: Different methods to adjust size

We can find in the table that the accuracy of method weighted mean is the best which reach 83.3567. And the method of best angle is also a good way to adjust size. According to the result we get, We finally use a weighted average of the size of the label of the extracted same trajectory. The longer the length of the edge is, the greater the weight is. And the result is shown in the Figure 10. We can find that it can obtain more accurate original size of the car.

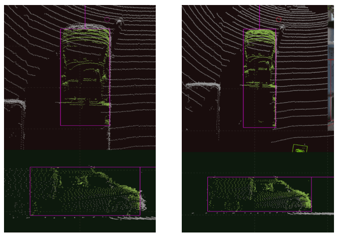

(a) before (b) After

Figure 10: Size Adjustment

4.3 In-trajectory Complement

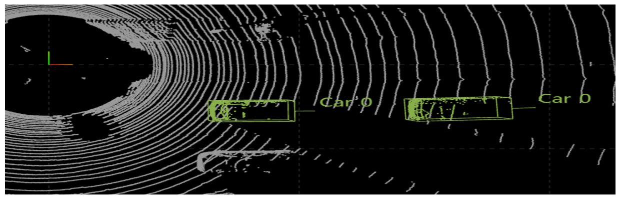
We judge whether there is a car that has been deleted or not or has not been detected by the position of the prediction box through the existing frames in the trajectory, and complete it. We tested 3272 trajectories in 20 scenes of kitti. The following results are obtained, and the comparison between the results before and after the frame completion is shown in Figure 11.

4.4 Pose adjustment

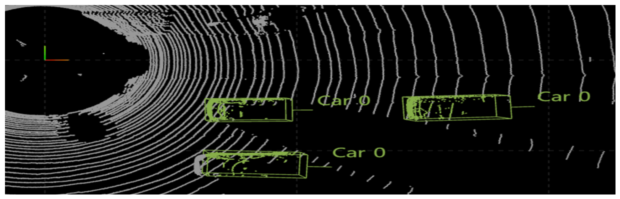
We try three decoding headers, single header, multiple headers and attention headers mentioned above.

Header	Rot. Mean Error (rad)	Tra. Mean Error (cm)		Accuracy (%)		Recall (%)	
		X	Y	pos.	neg.	pos.	neg.
Single	0.0514	8.45	9.48	0.996	0.99	0.99	0.996
Multiple	0.123	5.57	5.85	0.271	0.505	0.001	0.996
Multiple-Attention	0.0803	5.4	3.83	0.936	0.508	0.0673	0.995

Table 2: Evaluation of different headers



(a) Original scene



(b) After In-trajectory complement

Figure 11: In-trajectory complement

We train our network for 200 epochs for each header and test them separately. Several metrics are used to evaluate their performance and results are in Table 2.

What we find is that single-header method has perfect accuracy and recall of both positive and negative samples while others' are very poor. Single-header method has best adjustment on angel as well. However, the last two methods have better performance on translation than that of the first one and we discover that attention mechanism indeed improve performance of network compared to the one without attention. Unfortunately, recall of positive samples is so low in the last two methods that it is likely to recognize most of positive samples as negative samples. Therefore, we choose single-header method as our final decision.

5 Conclusion

In this project, we proposed an automatic labeling method of 3D point cloud. First, target detection and tracking are carried out for point cloud, and bounding boxes with low confidence are removed. Then, we select point clouds with the best viewing angle to determine the bounding box's length, width and height. Next, we use exponential smoothing to predict and merge the trajectory, and use neural network to determine whether the predicted points have objects, as well as fine-tuning the positions and angles of bounding boxes. If there is an object, we will continue to predict bounding boxes until the box doesn't have object. Finally, two trajectories overlapping each other will be merged.

According to our experiments, our method can complement some frames to a degree. However, our approach's performance is affected easily by performance of object detection and tracking models. If detection model doesn't work well and many bounding boxes are missing or labelled incorrectly, tracking model will generate lots of fragmentary trajectories. Those fragmentary trajectories will affect our prediction and thus make great impacts on performance of our network and trajectories merge. In addition, threshold of confidence is also hard to determine which should be adjusted under different circumstances. As for proposed network, we think if we select and design a more suitable loss function, problems about recall can be ameliorated.

References

[1] Luca Bertinetto et al. "Fully-Convolutional Siamese Networks for Object Tracking". In: *CoRR* abs/1606.09549 (2016). arXiv: `1606.09549`. URL: `http://arxiv.org/abs/1606.09549`.

[2] Charles R Qi et al. "Offboard 3d object detection from point cloud sequences". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 6134–6144.

[3] Charles R. Qi et al. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. 2017. arXiv: `1612.00593 [cs.CV]`.

[4] Shaoshuai Shi et al. *PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection*. 2021. arXiv: `1912.13192 [cs.CV]`.

[5] Xinshuo Weng et al. "3D Multi-Object Tracking: A Baseline and New Evaluation Metrics". In: *IROS* (2020).

[6] Bin Yang et al. "Auto4D: Learning to Label 4D Objects from Sequential Point Clouds". In: *CoRR* abs/2101.06586 (2021). arXiv: `2101.06586`. URL: `https://arxiv.org/abs/2101.06586`.