Joint Detection and Tracking for Automatic Annotation of 3D AD Data

LingRui Zhang ShuHeng Zhang RuiHao Zhang

Nov 2021

1 Introduction and Problems

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.

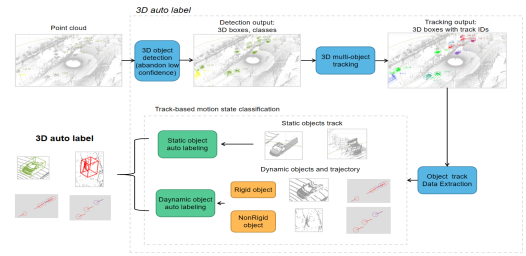
Currently, most 3D perception research has been focusing on real-time use cases and only considers sensor input from the current frame or a few history frames. Those models are sub-optimal for automatic labeling where the best perception quality is needed. However, there are several problems troubling 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 bboxees 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 bboxes to alleviate jitter problem and design a method to solve occlusion problem.

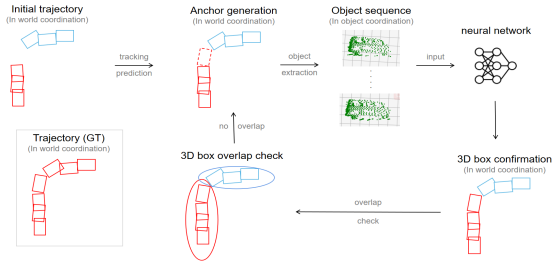
To solve such problems, our project can be divided into 2 stages (Figure.).

In the first stage, we finish object detection and object tracking and combined them to obtain an initial result where different objects are tagged different trajectory IDs. What's more, we use our algorithm to determine different objects' size and make it fixed trying to solving jotter problem.

In the second stage, we focus on refining object detection with assistance of object tracking. In details, for a certain object, we will use previous frame in its trajectory to generate anchor for next position and after applying neural network on it, we can learn that whether such frame misses the object and refine anchor if the object indeed exists. Then, we will iterate the process until there is no object in anchor or current trajectory overlaps other trajectory in this case the two trajectory belongs to the same object and they can be incorporated.



(a) Stage 1



(b) Stage 2

Figure 1: Approach

2 Method

2.1 Object Detection

The first step in the first stage is to get result of object detection. We adopted PV-RCNN [3] as our detection model, and leveraged kitti's data of object detection and tracking to train and test our model separately. Limited by server's resource, we only trained model with one-tenth data and then testing data is applied on it. The following is AP of different categories in validation and testing data.

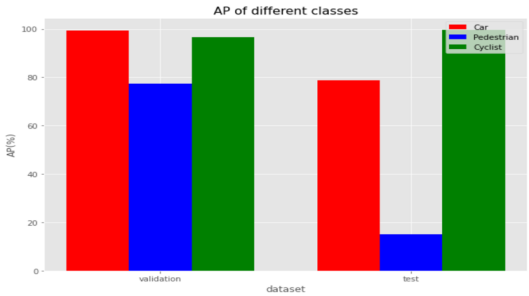


Figure 2: AP in different data sets

According to graph above, AP in validation set is excellent and in testing set, AP of car and cyclist is high but that of pedestrian is very low. Since we mainly care about cars, low accuracy of non-rigid objects doesn't impact us much, therefore we still adopt this model which will be used to give tracking model results of object detection. Considering that only small parts of data is used, we suppose current AP isn't very accurate and in the future we will take more data to train and test.

2.2 Object Tracking

After getting the result of target detection, we will track the target. Target detection is to detect a cuboid target frame from 3D laser point cloud, while target tracking is to give each cuboid target frame a unique ID, as the unique identification of the target object.

The method we use for target tracking is AB3DMOT[4]. AB3DMOT is a classical algorithm for 3D tracking. The idea of AB3DMOT algorithm is similar to the tracking algorithm Sort[1] in 2D domain. It also simply combines two components – Hungarian algorithm and Kalman filter algorithm. The simplest combination is used to achieve higher computational efficiency without large precision loss. According to the authors, the algorithm achieves a speed of 207.4FPS on KITTI data sets. AB3DMOT algorithm is not a thorough innovation of 3D tracking algorithm, but an extension of SORT algorithm in 3D field.

Using this algorithm to process the previously obtained target detection results, we get good tracking results, as shown in Figure 3. But there are some error cases, as shown in Figure 4(a). There are no people or cars in these places, but because of the limited accuracy of target detection, there are false positives. Therefore, we remove the results with low confidence before tracking. As shown in Figure 4(b), the problem of false positives is well solved. In the next steps, we will describe how to resolve the possible false negative problems that arise from this.

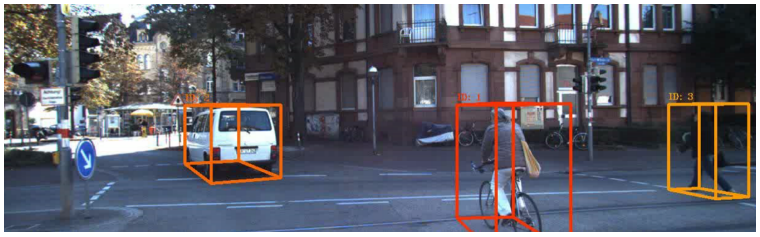


Figure 3: tracking result

(a) Some error cases (b) After removing results with low confidence

Figure 4: Caption

2.3 Preliminary Optimization of Size and Position

In point cloud detection, the object we detect is affected by angle and distance. We cannot 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.

In this section, We will frame-by-frame detect objections with the same ID by processing the tracked results. And according to the characteristics of point cloud detection, as shown in Figure 5, use its alpha, theta, and rotation y to calculate its best height and width.

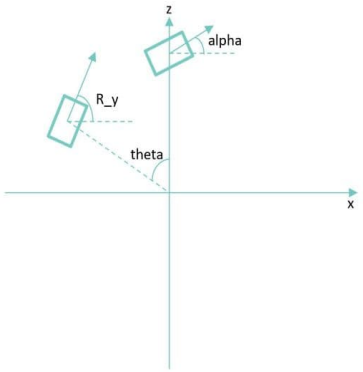


Figure 5: relation

We calculate theta = rotation y-alpha.We map theta and alpha to the sin function, and take their absolute values. And as shown in Figure 6, we get the most possible length and width in the limit case.

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

At the same time, we will select a corner where the point cloud is denser, and fit it with the label corner of the exact size we get like Figure 7. It can ensures that the center of the new label we get is the center of the actual object.[5] It

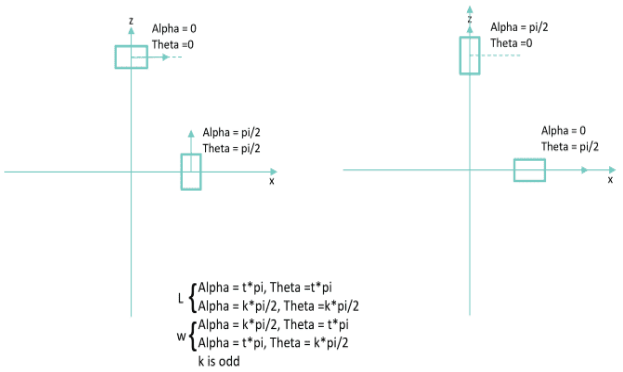


Figure 6: Most width and length

can also reduce the error caused by recalculating the center point.

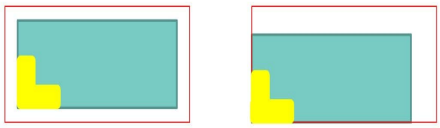


Figure 7: Fit box

We can see the label obtained from the original tracking in Figure 9, and the results after changing the size in Figure 10. Can surface after changing the size of our accuracy has improved.

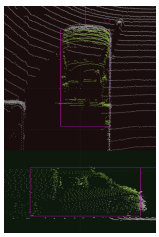


Figure 9: before

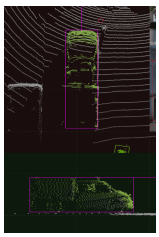


Figure 10: after

2.4 Anchor Prediction

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. Our method is to continue to predict one frame for each track according to the track information, and then check whether the predicted position still has a car or coincides

with another track. If there is a car, the position of the car is added to the current trajectory and the next frame continues to be predicted until there is no car or it coincides with another trajectory. If it is found to coincide with another track, merge the ids of the two tracks into one.

In predicting the position of the next frame, we use cubic exponential smoothing. As shown in Figure 10, given the information of the first 7 frames, we can predict the approximate position of the next frame through a smooth curve.

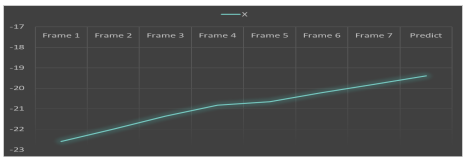


Figure 10: Exponential smoothing

The method to determine whether there is a car at the predicted location and whether it coincides with other tracks will be introduced below.

2.5 Anchor Refinement

Next step is to leverage network to refine the prediction. Structure of our network (Figure.11) uses that of PointNet ([2]) for reference. In our network, there are 2 point clouds inputted, one is from anchor predicted above and the other is in the same object's bbox in previous frame. Output is adjustment of position, biases of three axis (x, y, z), and adjustment of angle which ensures that objects are in the center of their own bboxes and directions of objects and bboxes are the same. We retain encoding part of PointNet and both point clouds will be encoded to a vector with 1024 features, after encoding, we trained two MLPs for classification and adjustment separately where the former judges whether there is an object in anchor and the latter outputs four adjustments. In order to compress model, both MLPs use the same encoding network.

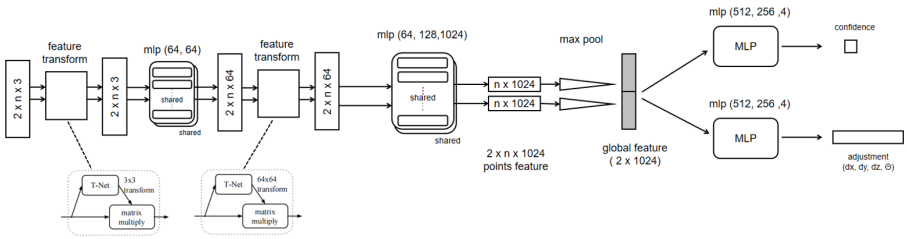


Figure 11: Caption

2.5.1 Data Preparation

We construct our training data from our own tracking result and Kitti dataset.

Training data requires objects to be adjusted and labels.To find what bias is the best, we calculated the error between the predicted results and the actual results. As shown in Figure 12, if the results obtained from the previous detection and tracking are used for prediction, there is still some error in the obtained results. However, if kitti's data were used, we found that there was almost no error in the results. This shows that our exponential smoothing prediction accuracy is very high, but the results obtained by our own program are not very accurate. In addition, we also found that the prediction of pedestrians was very inaccurate and difficult. Therefore, at the current stage, we first focused on the prediction of vehicles.

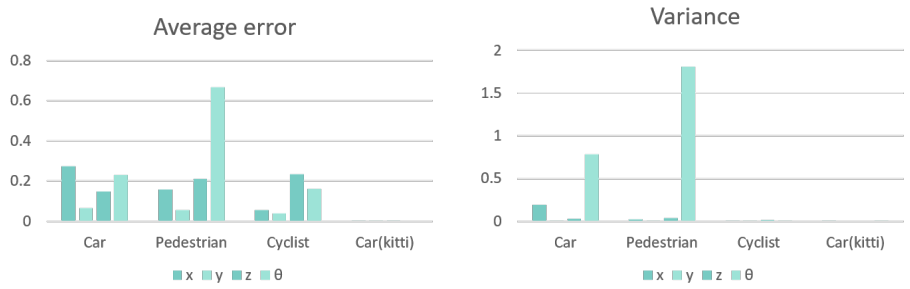


Figure 12: Prediction error

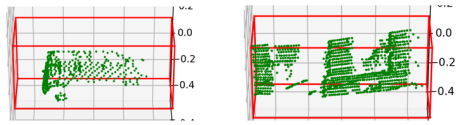
Input data are two point clouds which should be extracted directly from the point cloud of whole scene. According to kitti's data set format, we know the bottom center coordinates of cuboids, the length, width and height of cuboids, and the rotation angle of cuboids in horizontal plane. Since the cuboid must be horizontal, we can first judge whether the point is inside the rectangle on the base of the cuboid on the 2D projection.

Since we know the center coordinates of the rectangle and the rotation angle of the rectangle, we can easily obtain the coordinates of the four vertices of the rectangle. Then, through simple vector product, we can determine whether the point is in the middle of two sets of parallel lines, and finally determine whether the height is within the range of cuboid.

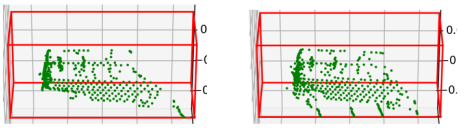
This process seems simple enough, but there is one very important step left to complete. Since the coordinates of points are in the radar coordinate system, all information of cuboid is described in the camera coordinate system. So, we need to unify the coordinate system. Fortunately, the Kitti dataset provides us with a matrix for transformation in the calib file, and we just need to understand what the matrix actually means to convert coordinates correctly.

Considering there may be no object in anchor, we construct two kind of inputs. Negative data has no object in anchor while positive data has object

(Figure.13). In addition, we augment data through sampling two point clouds in non-adjacent frames.



(a) Negative data



(b) Positive data

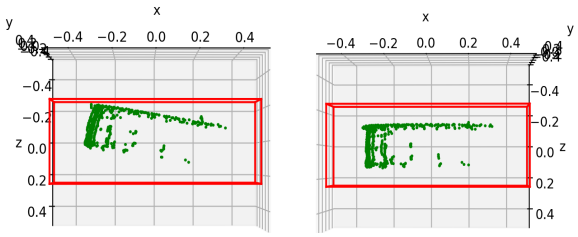
Figure 13: Training data

2.5.2 Model training

We get following statistic (Figure.15) after applying over two thousand samples training model with 50 epochs.

Training statistic records real-time changes of loss and accuracy on training dataset. It is obvious that in the end of training, our model can perfectly judge whether there is an object and adjustment of pose isn't bad as well which only has millimeter-level error.

Statistic on validation data is also good. Loss of adjustment can get centimeter-level. Classification is still perfect and relevant statistic (recall, specificity and so on) reach maximum value as well.

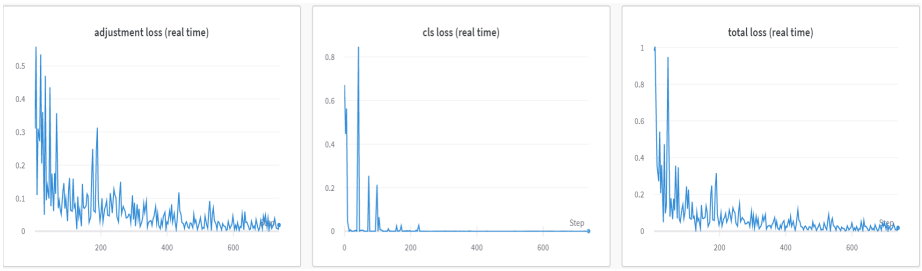


(a) Original pose (b) Refined pose

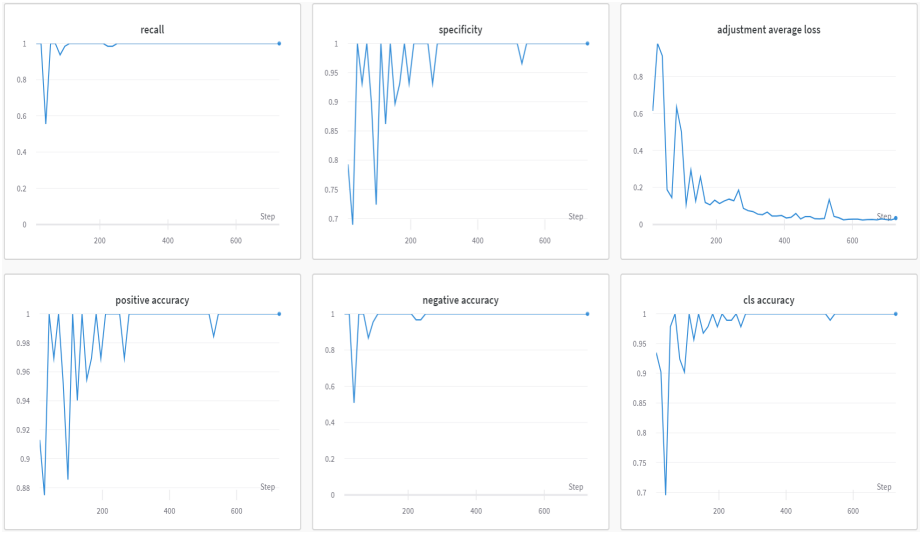
Figure 14: Visualized result

2.5.3 Result

Apply trained model on testing point cloud. From bird's eye view the data is like Figure.14 (a). Output of our model is Figure.14 (b) and it is very excellent.



(a) training Statistic



(b) Statistic on validation data

Figure 15: Statistic during training

2.6 Trajectory incorporation

After we get the adjusted position, we are going to check all object and combine different object if they are exactly same object. We use the method of prediction to get the next stage of box. Then if another box from another ID has the similar length and width with predict one. Show in Figure 16 Because there is little difference in the height of the car on the actual road, we mainly use the overlooking angle, using the frame length and width is enough. Therefore, we fix the point in the upper left corner and calculate their overlap IOU. We will incorporate these two trajectories by their frame if overlap IOU lager than the threshold we make. If there are no points in our predicted box, We continue to predict two times.

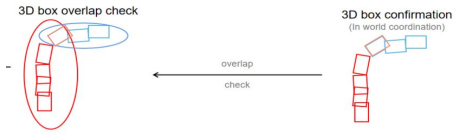


Figure 16: Trajectory incorporation

3 Summary

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.

For future work, we will also add more types, such as trucks, roadblocks, etc. At the same time, the labeling of non-rigid objects such as pedestrians will also be considered, focusing on solving the low accuracy problem caused by the direction.

References

[1] Alex Bewley et al. "Simple online and realtime tracking". In: *2016 IEEE international conference on image processing (ICIP)*. IEEE. 2016, pp. 3464–3468.

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

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

[4] Xinshuo Weng et al. "3d multi-object tracking: A baseline and new evaluation metrics". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 10359–10366.

[5] Bin Yang et al. "Auto4D: Learning to Label 4D Objects from Sequential Point Clouds". In: 2021. arXiv: `2101.06586 [cs.CV]`.