Canonical Transformation and Data Augmentation for Enhanced Box Refinement in a Two-Stage 3D Object Detection Pipeline

Matthew Hanlon
ETH Zurich

Rong Zou
ETH Zurich

Abstract

In this work, we build upon the existing two-stage 3D object detection pipeline for the task of detecting cars in LiDAR data. We incorporate two modules from the existing literature into the box refinement stage of the baseline network and study their effect on the overall performance. With the canonical transformation module, we introduce a step where the pooled input points are transformed into a corresponding local coordinate frame before being fed into the network. We additionally introduce more variance into the training set by randomly augmenting the samples with geometric transformations. We show that a significant increase in performance can be achieved using the canonical transformation module and that while the data augmentation method provides an improvement over the baseline network on its own, the effect is negated when used in combination with the canonical transformation mechanism.

1. Introduction

After training the baseline network, it becomes clear that there are several aspects that could be improved. Firstly, while the baseline network can accurately regress bounding box parameters for object detections at short distances, it seems to struggle with those that are further away, reducing the overall mAP scores. Furthermore, when reviewing the training scheme of the baseline implementation, we note that none of the training data augmentation methods commonly used for image data are applied to the training samples, which may increase the likelihood of the model overfitting on the training set.

With the goal of overcoming these shortcomings of the baseline model and improving the prediction accuracy, some modifications are made on the basis of the preliminary network. Specifically, in order to enable the network to make more accurate predictions for distant objects, we implement an approach that uses more information from the first stage predictions, transforming the pooled input points of each prediction into their corresponding canonical coordinate frames. This way, instead of regressing the bounding box parameters directly, our network learns to regress only the differences between the the first stage predictions and their corresponding ground truths, which can then be added back to the first stage predictions to recover the full bounding box parameters. Besides, to make the model more robust and reduce the risk of overfitting, we introduce more variations into the training set by augmenting the samples with randomized geometric transformations.

2. Related Work

Both aspects of the approach described in section 1 are inspired by the work of [1], where a very similar two-stage object detection pipeline was implemented.

The authors of [1] claim that the canonical transformation of the pooled input points enables the network to learn better local spatial features, while taking advantage of the high recall first-stage predictions. This is due to the fact that after transformation, the input points of each proposal lie in the local coordinate frame of the corresponding first-stage prediction. In this way, for each positive sample, the geometric features of the object have similar orientations and locations. This lowered variance of the appearance of positive samples aims to improve the capability of the network to identify them. By transforming the ground truths in the same way and forming the training pairs, this approach also simplifies the regression task, as the network only needs to learn to predict the bounding box residuals between the prediction and the ground truth, instead of regressing the full bounding box parameters for each prediction.

A common technique to avoid overfitting in image classification tasks is to make use of training data augmentations [2]. Introducing random transformations to the training data increases the variance between the samples. Several of the data augmentation methods used for 2D images are not applicable for our problem. However, several common geometric transformation techniques generalize quite easily to data that is in a 3D space.

Although the authors of [1] made use of both of these approaches in their implementation, they did not isolate the effects of these methods. In our work, we aim to study their effects in an isolated fashion to determine the benefits and/or drawbacks of using these modules individually and in combination for the task of 3D object detection.

3. Method

As mentioned before, both aspects of our approach were also implemented by [1]. Parts of the implementation were adapted from their released code[1] for our project.

3.1. Canonical Transformation

3.1.1 Basic idea

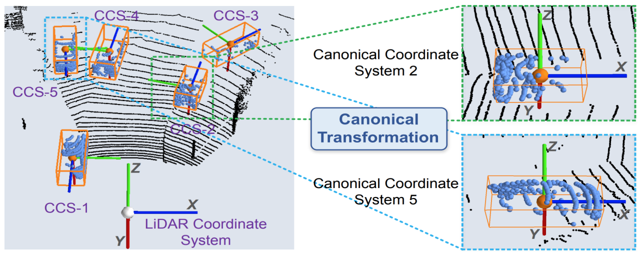


Figure 1. Visualization of the canonical transformation process. The pooled input points are transformed to the local coordinate frame of the predicted bounding box. CCS denotes the canonical coordinate system. [1]

The visualization of the canonical transformation process is shown in Fig. 1. For each 3D proposal that consists of a predicted bounding box and a set of pooled points expressed in world coordinates, we transform the position vectors of the point cloud relative to the box center from the world coordinate system to the canonical coordinate system by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{p,canonical} = R_{CW} * (\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{p,world} - \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{b,world})$$

where $[x, y, z]^T$ denotes position, the first subscript denotes pooled points (p) or center of the stage-1 predicted box (b), the second subscript means the coordinate system that the position is expressed in, and R_{CW} is the rotation matrix from world to canonical coordinate system, which is

$$R_{CW} = \begin{bmatrix} cos(rot_y) & 0 & \text{-}sin(rot_y) \\ 0 & 1 & 0 \\ sin(rot_y) & 0 & cos(rot_y) \end{bmatrix}$$

in which rot_y is the orientation of the predicted box (only consider the rotation around y).

[1]https://github.com/sshaoshuai/PointRCNN

3.1.2 Treatment for different sets

For the training set and validation set, canonical transformation is performed not only on the 3D proposals from the first stage, but also on the ground-truth bounding boxes corresponding to the proposals, i.e. their labels.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{g,canonical} = R_{CW} * (\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{g,world} - \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{b,world})$$

$$\theta_{g,canonical} = \theta_{g,world} - rot_y$$

where θ denotes the orientation and the subscript g the ground-truth box. In this way, the loss function computes the loss using the transformed predictions and labels.

Besides, for the validation set, after calculating the loss the refined predictions are restored to the world coordinate system using inverse canonical transformations. This is for the convenience of visual validation.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{r,world} = R_{WC} * \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{r,canonical} + \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{b,world}$$

$$\theta_{r,world} = \theta_{r,canonical} + rot_y$$

in which the subscript r denotes the refined bounding box (stage-2 output), R_{WC} is the inverse of R_{CW},

$$R_{WC} = \begin{bmatrix} cos(rot_y) & 0 & sin(rot_y) \\ 0 & 1 & 0 \\ \text{-}sin(rot_y) & 0 & cos(rot_y) \end{bmatrix}$$

For the test set, we canonically transform the first stage proposals for box refinement and perform inverse transformation after obtaining the refined boxes, so that the final output of the network is a set of refined boxes in the world coordinate system.

3.1.3 Feature augmentation

The advantage of using canonical coordinates is that it enables robust learning of local spatial features. However, compared to world coordinates, depth information is discarded in local coordinate system. To compensate for this, the normalized distance to the LiDAR, which contains the depth information, is included in the feature vector of each point. For a point $[x, y, z]^T$, the normalized depth is calculated as

$$d = \frac{\sqrt{x^2 + y^2 + z^2}}{m} - 0.5$$

in which m is a value close to the range of the LiDAR, in our experiments it is set to 70 meters.

In addition, the laser reflection intensity of each point is also added into its feature, since it may be helpful for the classification task.

3.2. Training Data Augmentation

When augmenting the training samples, the modality of the data limits the possible augmentations to those of a geometric nature. We implemented three types of geometric transformation that can be applied to the samples, with the goal of increasing the variance and thus improve the performance on the test and validation sets.

The augmentations are applied to the training samples at random. The probability with which each type of augmentation is applied to a given sample can be configured by the user. For the rotation augmentation, the user can also set the range of angles by which a sample can be rotated.

3.2.1 Rotation

The rotational augmentation is applied by rotating the sample points, the first stage prediction box and the ground truth box around the y-axis of the global coordinate frame by a randomly sampled angle. The range from which the angle is sampled can be determined with the configuration file. For the experiments where the training data was augmented, the angles were sampled from the range $[-10°, 10°]$ and the augmentation was applied to all samples ($p_{rot} = 1$). This process introduces more positional variance of the objects into the train set. An example of the rotational augmentation can be seen in Fig. 2.

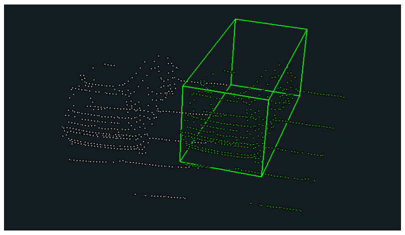


Figure 2. Training sample after a rotational augmentation has been applied. Original sample points in white, augmented sample in green.

3.2.2 Scale

The scaling augmentation is applied by multiplying the coordinates of all sample points and all of the bounding box parameters except for rot_y by a value randomly sampled from a given range. For the experiments the scale value is sampled from $[0.95, 1.05]$ and applied to all training samples ($p_{scale} = 1$).

3.2.3 Flip

The flip augmentation works by multiplying the x-coordinate of all the sample points by -1. The x parameter of the ground truth and first stage prediction boxes are also multiplied by -1 and the rot_y parameters are also adjusted accordingly. For the experiments, this augmentation was applied randomly to roughly half of the training samples ($p_{flip} = 0.5$). Fig. 3 presents an example of the flip augmentation.

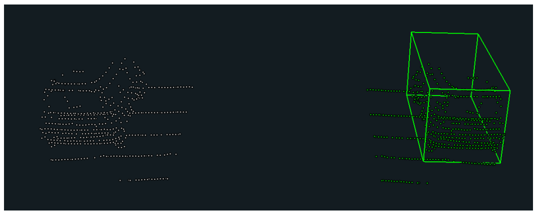


Figure 3. Training sample after a flip augmentation has been applied. Original sample points in white, augmented sample in green.

4. Results & Discussion

4.1. Mean Average Precision (mAP)

The validation scores of the baseline and improved networks after training for 35 epochs are shown in Tab. 1. The values in the table are mAPs for different difficulty categories, as well as the largest improvement over the baseline achieved by the top-performing network in that category.

Results on Validation Set after 35 Epochs			
Networks	Easy	Moderate	Hard
Baseline	77.69	71.24	64.57
Data Aug.	84.85	74.78	73.14
Can. Trans.	94.73	**89.33**	**88.48**
Data Aug. + Can. Trans.	**94.93**	89.12	88.29
Relative Improvement	17.24	18.09	23.91

Table 1. Comparison of validation set performance for the different networks.

As can be seen from Tab. 1, both modules described in Sec. 3 individually improve the network performance in all three difficulty categories, with canonical transformation achieving a significantly greater improvement. For the network with training data augmentation, compared to the scores of the baseline model, the largest increase is achieved in the hard category (8.57%), while the moderate class shows the smallest improvement (3.54%). Using the canonical transformation results in more considerable improvements, with increases of respectively 17.04%, 18.09%, and 23.91% for the three categories over the baseline network. When combining both modules in one network, the performance is quite similar to the one that only uses canonical transformation.

To illustrate the above discussion more clearly and in detail, example predictions for the same scene using the baseline network and improved networks are shown in Fig. 4.

In Fig. 4a it can be seen that the baseline network accurately predicts the locations of the two objects closest to the sensor with an IoU with ground truth > 80. It is however noticeable that the IoU with ground truth objects decreases with the distance to the sensor, with lowest value being 65. All of the detections made by the baseline network in this scene are correct.

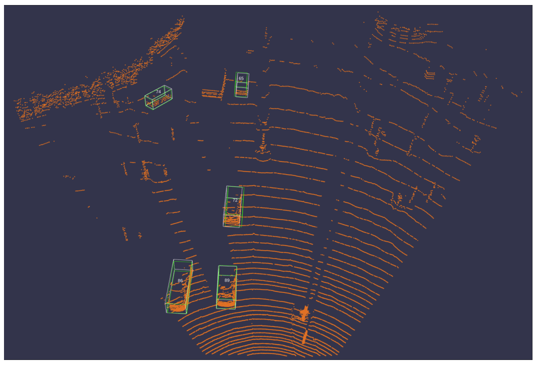
In Fig. 4b we can see that for this scene the overall results obtained using the network with data augmentation are similar to those obtained by the baseline network. The predictions have a similar accuracy at close and far ranges, again the IoU scores are higher for objects close to the sensor and lower for the distant ones.

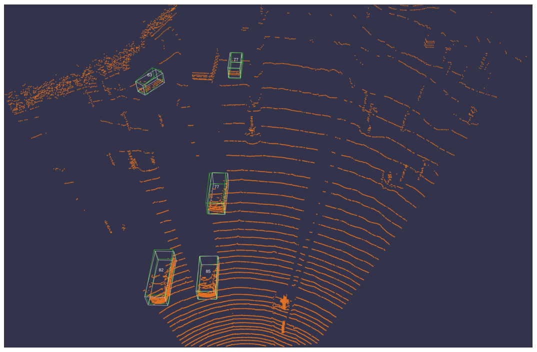
We can see in Fig. 4c that the results obtained by the network that was trained using the canonical transformation are much more accurate than those obtained by the baseline network. The IoU for all correct predictions is > 80, even for the object that is furthest away from the LiDAR, with the lowest being 81. This in particular shows that the use of canonical transformation increases the detection accuracy at greater distances compared to the baseline network and the one trained with data augmentation. However, in contrast to the two previous networks, this network makes an incorrect bounding box prediction for a cluster located near the boundary of the point cloud.

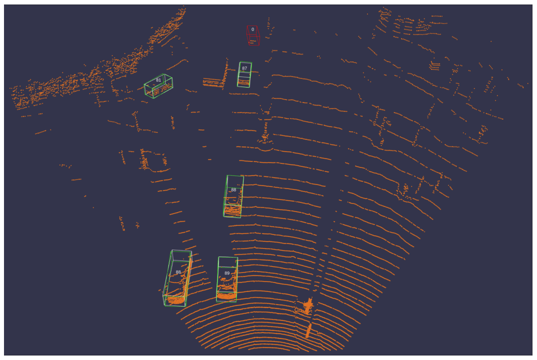
Fig. 4d shows very similar results to those in Fig. 4c. The predictions made by the network trained with both data augmentation and canonical transformation obtain very similar IoU scores in this scene. The network also falsely classifies the same cluster close to the maximum range of the LiDAR and therefore makes a false prediction.

The networks that include the canonical transformation module in their training perform significantly better than those that do not. However, this module may also make the network more susceptible to false detections far away from the origin. In the context of autonomous driving, the consequences of false detections are less severe than missing an object, especially if the only false detections are far away. This weakness of the canonical transformation module most likely comes from the loss of depth information that it incurs. A possible solution could be incorporate the distance to the sensor more heavily in the feature vector of the input points, for example by feeding the augmented features described in Sec. 3.1.3 through an additional convolutional network. Another possible reason is that the value of the hyperparameter m used in Sec. 3.1.3 is inaccurately selected, therefore better results may be obtained by increasing m.

It should also be noted that, although both modifications can individually improve the baseline network performance, the network that includes both modules has almost identical


(a) Baseline network


(b) Training with data augmentation


(c) Training with canonical transform

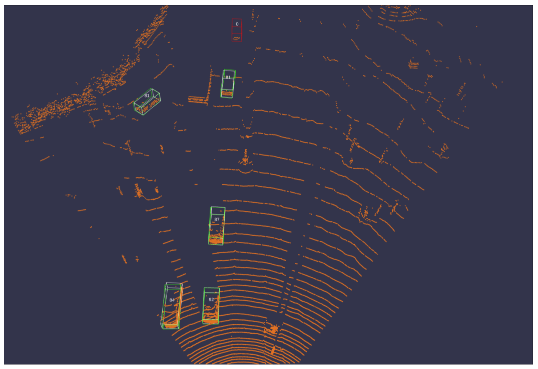

(d) Training with data augmentation and canonical transform

Figure 4. Visualizations of detections for the same scene using different networks after training for 35 epochs.

performance to the network using only the canonical transformation, with the difference in mAP being merely around 0.2% for all categories. This is most likely due to the fact that the intended purpose of the canonical transformation module is to negate the effects of geometric variations of the positive samples. Thus, the introduction of this type of variance through the data augmentation module has little to no effect on the results. From this result we can conclude that the use of geometric data augmentation techniques are not of any benefit when using a canonical transform module and it is better to save the computational effort of augmenting the training samples.

4.2. Convergence Speed

Another aspect worth noting is the speed of convergence of the training. In Fig. 5 we can see how the mAP for the moderate category increases during training for each of the networks (the graphs for the other categories follow an almost identical trend). It is clear to see that the networks that include the canonical transformation module converge much faster than the baseline network and the network trained with only the data augmentation module. After as little as 5 epochs, the training of the networks using canonical transformation have already converged, whereas the mAP curves of their counterparts gradually increase, showing no sign of convergence even at the end of training. Further evidence to support this, is that the results on the Codalab grader (test set) were almost identical after 5 epochs as after 35 epochs for the networks incorporating canonical transformation.

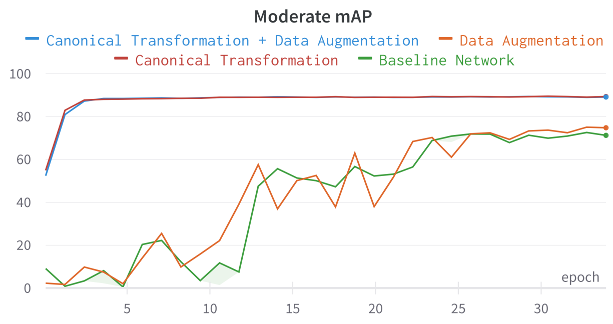


Figure 5. Evolution of moderate mAP during training.

4.3. Resource Consumption

In terms of the GPU memory consumption, the process GPU memory allocated for all the networks are similar, which is around 70%. While it is hard to determine the exact required training runtime for the networks due to the unstable AWS instances used, we did not notice a significant difference between the training time of the networks when taking into account all 35 epochs of training without regard to convergence.

4.4. Final Grader Results

The best results on the test set using the CodaLab grader were achieved by the network using the canonical transform module, after only five training epochs. These results can be seen in Tab. 2.

Grader Test Metrics		
Easy mAP	Moderate mAP	Hard mAP
94.76	**89.23**	**87.00**

Table 2. Grader performance of the network using the canonical transform module after training for 5 epochs.

5. Conclusion

In this report, we have presented two methods for addressing the shortcomings and improving the performance of the baseline network, namely, canonical transformation for the model to predict distant objects more accurately, and data augmentation for it to not overfit the training data. We also studied their respective effects on the baseline network and the effect when they are combined. It is shown through experiments that, by applying the presented modules, compared to the baseline model, the mAP metrics improve remarkably by 17.24%, 18.08% and 23.91% on the validation sets of three different difficulty levels, respectively. Experimental results also show that the separate application of both modules can achieve significant improvements over the baseline model, while their combined application provides a similar performance as applying the canonical transformation alone. In other words, the data augmentation module is redundant when canonical transformation is performed in the network. Moreover, we find that by utilizing canonical transformation in the model, the increased performance can be achieved with a considerably reduced training time, meaning that it leads to much faster convergence. On the other hand, the canonical transformation can cause the model to produce false predictions near the boundary of the point cloud, but adjusting parameters and hyperparameters related to the depth feature of the points may alleviate this adverse effect.

References

[1] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud, 2018. 1, 2

[2] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, Jul 2019. 1