Fast 3D Object Detection with RGB-D Images Using Graph Convolutional Network

[1]Masahiro Takahashi, [1]Takumi Kitsukawa, [2]Alessandro Moro, and [1]Kazunori Umeda

***Abstract*— In this paper, a lightweight 3D object detection model using color and depth images is proposed. In recent years, several studies have focused on the application of deep learning to object detection. They use many techniques, including improved feature extraction methods and instance segmentation, to increase the accuracy. However, such 2D object detection has its limitations. Other models and methods are needed to deal with occlusion and to identify 3D positions. In contrast, there have been many studies in this field applying deep learning to 3D object detection. However, many of them are computationally expensive and difficult to run in real time because they deal with dense point clouds. In the proposed model, after feature extraction from the color image, a sparse point cloud is created from the range image to achieve fast object detection. Graph convolution for point clouds and feature extraction with depth information are also used. As a result, the proposed model achieved 56.4 fps when using ResNet34.**

I. Introduction

Detecting and recognizing objects can be applied to various fields such as security and marketing. Therefore, it is important to be able to perform these tasks fast and accurately because you are able to improve the performance of the whole camera system. In addition, sensors that can also measure depth images, such as stereo cameras, are becoming cheaper and more readily available.

In recent years, many methods based on deep learning have been proposed in this field. A typical example of a method for object detection is Mask R-CNN [1] by He et al. In R-CNN systems [2, 3, 4] such as Mask R-CNN, object detection and classification are performed using a network structure called a Region Proposal Network (RPN). In Mask R-CNN, a network—for instance, segmentation—is added to the basic structure, such as RPN, in order to extract the position of objects in the image more accurately. However, this method has the problem of extremely slow processing speed. This is because the network for locating object regions and the network for classifying them are defined as separate models. As mentioned above, Mask R-CNN has an additional network structure, so even in an optimized environment, the speed is about 5 fps, which is not sufficiently real time.

Redmon et al. proposed You Only Look Once (YOLO) [5] as a fast object detection method using deep learning. YOLO has been updated several times [6, 7]; the latest model is YOLOv4 [8]. YOLO uses unified detection, which enables fast detection. In YOLOv4, feature pyramid network (FPN) [9] is added as a neck between the backbone network, which extracts features, and the head part, which outputs object positions and other information. This improves the performance of object detection for objects of different scales. However, it has been pointed out that this method is vulnerable to occlusion between objects. This is a problem shared by many object detection models for color images, such as YOLO.

In contrast, there have been many studies in this field that have applied deep learning to 3D object detection. Among them, PointNet [10] and PointNet++ [11] enable feature extraction from point clouds, and models such as VoteNet [12] and YOLO3D [13] have been proposed to apply these models to object detection and semantic segmentation tasks. However, most of these studies deal with dense point clouds, a task that is computationally expensive, difficult to operate in real time, and requires high graphic processing units (GPU) power. In addition, most assume the use of point clouds obtained by depth sensors such as LiDAR, are designed for large-scale environments, and cannot be easily implemented. Another lightweight model that uses RGB-D is Complex-YOLO [14]. However, this model also requires a sensor that can acquire a wide range of distances because it is based on the assumption that the point cloud from a bird's-eye view is used as an image. In addition, this model is not suitable for indoor environments where it is difficult to use such a sensor.

In our research, we propose a fast and lightweight 3D object detection model using color images and depth images obtained from an RGB-D camera. For the created point cloud, we further use Graph Convolutional Network (GCN) [15] and Graph Convolutional Network II (GCN II) [16] to perform feature extraction with depth information.

In this paper, we evaluate the object detection accuracy of the proposed model using the SUN RGB-D dataset [17], which is a dataset for 3D object detection. We also compare and evaluate the processing speed of each backbone network.

II. Network Architecture and Training

A. Network Architecture

The network architecture proposed in this study is shown in Fig. 1. The network can be roughly divided into three parts: the color feature extraction part (backbone network), the graph creation part, and the 3D feature extraction part.

Color feature extraction part: Using existing feature extraction models such as VGG16 [18] and ResNet [19], we extract features from color images. The high-dimensional features obtained by this method are used directly in the subsequent models to utilize the features from color images for 3D object detection, a process that is necessary for class classification and object detection. Since the purpose of this part is to use features obtained from general color images, the

[1]The Course of Precision Engineering, School of Science and Engineering, Chuo University, 1-13-27 Kasuga, Bunkyo-ku, Tokyo, Japan (e-mail: kitsukawa@sensor.mech.chuo-u.ac.jp)

[2]RITECS Inc., 3-5-11 Shibasaki, Tachikawa-shi, Tokyo, Japan

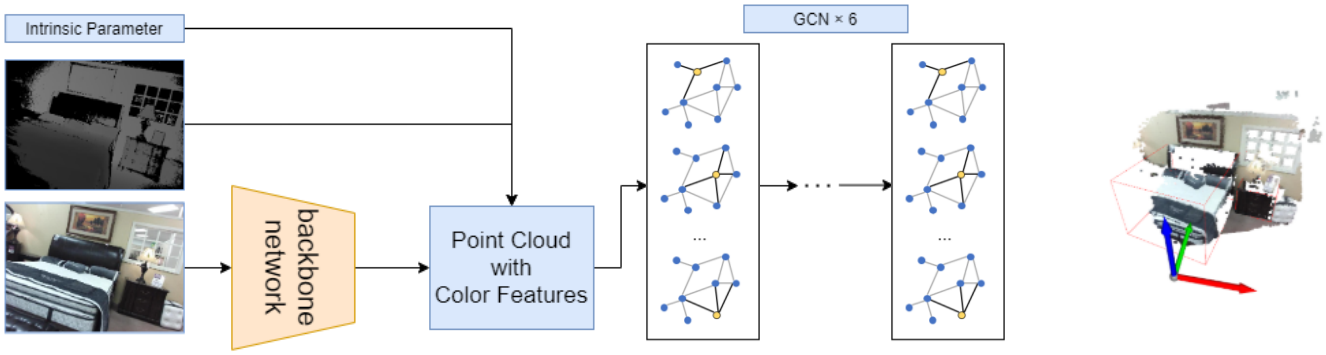

Fig. 1 Network architecture of the proposed model
Input: color, depth image, and intrinsic parameter; Output: 3D bounding boxes and classification

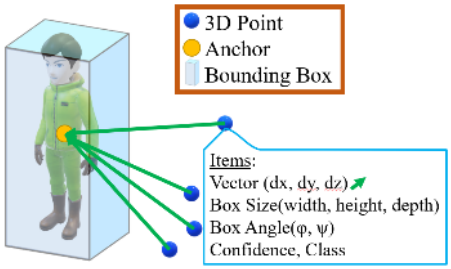

Fig. 2 Output of the proposed model

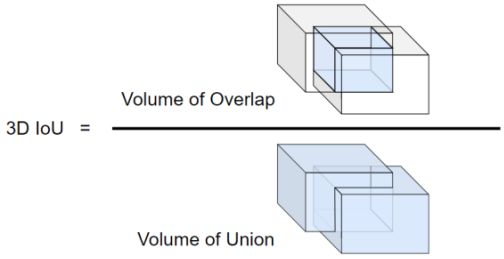

Fig. 3 Description of 3D IoU

same trained model as in the object detector [1–8, 20] can be used.

Graph creation part: First, the point cloud with attribute values of high-dimensional features obtained from color images is converted into a graph structure with edges for 16 neighborhoods. These edges are searched with K-Nearest Neighbor (KNN). At this time, a sparse point cloud is generated according to the feature map, and random sampling is performed from it. By doing this, we can fix the number of points to be input to subsequent models and obtain the same effect as data augmentation by random sampling.

3D feature extraction part: The point cloud is formed as a graph structure, and 2-layer GCN and 4-layer GCN II are used to extract features and estimate the bounding box, taking into account the positions of neighboring points. At that time, the 4-layer GCN II was placed between the 2-layer GCN. The reason for this arrangement is that the number of feature dimensions of GCN II cannot be changed due to its nature. GCN II takes the subtraction between input and output in the same way as the residual block in ResNet. Therefore, the number of feature dimensions of the input and output must be the same, and the number of feature dimensions cannot be changed. On the other hand, vanilla GCN can change the number of feature dimensions; thus, we sandwich GCN II with vanilla GCN. Since our model deals with the coordinates of the point cloud, there are negative values in the input. Therefore, TanhExp [21], which can handle negative values as well as Leaky ReLU [22], is used as the activation function. The output format is based on unified detection, a method common to YOLO [5–8] and single shot detection (SSD) [20]. Unified detection is a method that stores the coordinates of a bounding box and the result of classifying it in a single tensor, thus enabling simultaneous output of classifying and region identification. As shown in Fig. 2, in the proposed model, each 3D coordinate of the output is a vector from the point of interest to the center point of the bounding box (dx, dy, dz), the size of the bounding box (width, height, depth), the angle of the bounding box (φ, ψ), the confidence of the output vector, and the classification result. The output size is the batch size x the number of 3D points x (9 + number of classes). Finally, the bounding box information and the classification information are output together as a single tensor. Therefore, as compared to the case where the localization of the bounding box and the classification of the class are performed separately, the model is more compact without large branches, which enables faster object detection.

With the network structure described above, the proposed model outputs 3D bounding boxes from color and depth images.

B. *Training and Evaluation*

In the proposed model, the VGG and ResNet used in the backbone network are pre-trained with ImageNet [23]. This allows us to estimate 3D objects with a certain number of pre-defined feature regions in the image.

Since the output format of the proposed model is unified detection, a loss function similar to that of YOLO is also used for training. This loss function is defined as the sum of the mean squared error (MSE) loss of the vector toward the center coordinate of the bounding box, the MSE of the size of the bounding box, the MSE of the rotation angle of the bounding box, and the binary cross entropy (BCE) loss of the confidence of the vector. The loss function is:

The subscript “out” represents the output of the model, and “tar” represents the teacher data. $cos\theta$ is obtained from the angle θ between the vector going to the center point of the

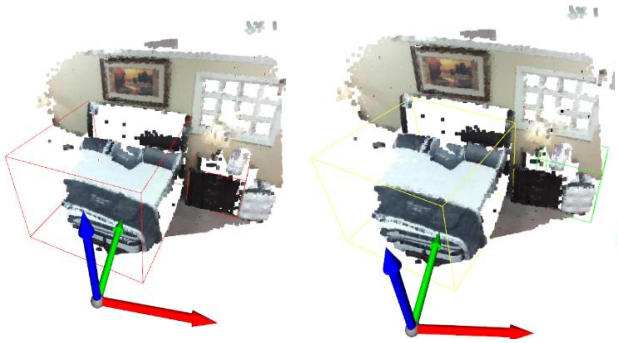

Fig. 4 Ground truth and model output of successful scene
Left: ground truth; Right: model output

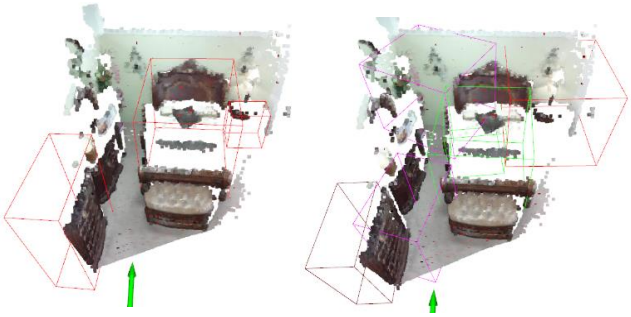

Fig. 5 Ground truth and model output of failure scene
Left: ground truth; Right: model output

bounding box and the vector of the correct answer. λ_{bb} and λ_{nobb} are the coefficients for the loss computed when the object is present or absent, respectively. In this case, we use λ_{bb}=1 and λ_{nobb}=10. The reason for setting λ_{nobb} large is that if we set it small, the loss to suppress false positives will become small, resulting in a large number of false positives. *cls* is the class classification result, which is output as a one-hot vector, so we calculate the loss by BCE.

A typical method of selecting the best bounding box from the obtained candidates is non-maximum suppression (NMS), which is used in R-CNN [2]. This method is used to eliminate bounding boxes estimated for the same object based on the

$$\begin{aligned} \text{Loss} = \\ \lambda_{bb}\ \{&\text{MSE}((dx,dy,dz)_{out}, (dx,dy,dz)_{tar}) \\ &+ \text{MSE}((width, height, depth)_{out}, (width, height, depth)_{tar}) \\ &+ \text{BCE}(cos\theta_{out}, cos\theta_{tar}) \\ &+ \text{BCE}(cls_{out}, cls_{tar})\} \\ + \lambda_{nob}\{&\text{BCE}(cos\theta_{out}, cos\theta_{tar})\} \end{aligned} \quad (1)$$

IoU (Intersection-over-Union), a score that indicates the degree of overlap between regions. Object detectors that handle 3D information, such as YOLO3D, calculate the IoU for two dimensions from two directions—the frontal direction and the vertical direction of the sensor—and select the bounding box by NMS. However, this method computes the IoU twice for the same 3D bounding box, which is inefficient for parallel computing on a GPU. In the proposed model, we define 3D IoU, which represents the degree of overlap of the volumes, as shown in Fig. 3, and perform NMS based on it. The proposed model defines 3D IoUs, which represent the degree of volume overlap, and executes NMS based on these 3D IoUs. The threshold value of IoU in NMS is 0.6, which is the best score in the proposed model, while 0.5 was used in YOLO and others.

III. Evaluation Results for Public Dataset

In this section, we describe our validation of the proposed model through multiple verifications. For the validation, we used the SUN RGB-D dataset [17], created by Song et al. It contains color images of 10,335 scenes, corresponding depth images, intrinsic parameters of the camera that captured the images, and annotation information, such as bounding boxes. The SUN RGB-D dataset was created based on three RGB-D datasets: NYU depth v2 [24], Berkeley B3DO [25], and SUN 3D [26].

A. Evaluation 1: Object Detection Accuracy

To verify the object detection accuracy, we trained 10 classes of the SUN RGB-D dataset: bed, table, sofa, chair, toilet, desk, dresser, nightstand, bookshelf, and bathtub for 100 epochs. The batch size was set to 3, the input image size was set to 224×224 pixels, and the learning coefficient was set to 0.0001. Adaptive moment estimation (Adam) was used as the optimization method.

Fig. 4 shows an example of successful object detection using ground truth and the proposed model. In this scene, bed and nightstand are given as correct answers, and we can say that both of them are detected accurately. In particular, for bed, the angle and size of the object are accurately detected, indicating that it is possible to detect even large objects. In the case of the nightstand, there is an error in estimating the center of the object, but the size of the object is correctly estimated.

Fig. 5 shows an example of a failure. In this scene, each object is detected, but there is a false positive in the middle of each object. Comparing these two examples, we can see that there is a difference in the quality of the point cloud. In the successful case, the point cloud obtained for each object has less noise, and the point cloud is coherent, which means that feature extraction by GCN that considers neighbor points is successful. On the other hand, in the failure case, the point cloud obtained for each object is not coherent, and there is a lot of variation in the point cloud, even in the area where there is no object.

Table I shows the average values of the 3D IoU results for different backbone networks. The results show that the best results are obtained when ResNet34 is used as the backbone. The model with layers deeper than those of ResNet34 did not work well because the size of the input image used in this study was too small, and the output size was insufficient. The 3D IoU score is above 0.5, which means that the model is able to detect objects with some accuracy; however, it is not as accurate as the VoteNet score of 0.83 as shown in Table II, which is the same model used for 3D object detection. The reason for this is that VoteNet uses a dense point cloud for detection, while the proposed model uses a sparse point cloud, which makes 3D estimation more difficult. However, as shown in Fig. 5, although there are many false positives, the proposed model is able to identify the object position; thus, it is possible to solve

TABLE I. 3D IOU SCORE BY BACKBONE

Backbone name	**Mean 3D IoU**
VGG16	0.586
ResNet18	0.603
ResNet34	0.627
ResNet50	0.606
ResNet101	0.610
ResNet152	0.581

TABLE II. COMPARISON WITH VOTENET

Model Name	**Mean 3D IoU**
VoteNet	0.831
Ours (ResNet34 + GCN II)	0.627

TABLE III. THE RESULTS OF SPEED TEST

Backbone name	**Speed [fps]**				**Standard deviation**
	max	*min*	*mean*	*median*	
VGG16	91.6	4.6	79.4	86.7	13.6
ResNet18	69.5	4.6	60.1	64.7	9.7
ResNet34	60.7	4.6	53.0	56.4	7.6
ResNet50	21.9	4.0	20.6	21.1	1.5
ResNet101	19.0	2.7	17.9	18.1	1.3
ResNet152	16.8	4.0	16.1	16.3	1.0

this problem by increasing the number of learning epochs and adjusting the learning coefficient.

B. Evaluation 2: Object Detection Accuracy

Next, we describe the verification of the processing speed of the proposed model. In order to verify the speed of the proposed model, a total of 200 frames were inferred, and the statistics were calculated using the results. Table III shows the results of the processing speed when the backbone network was changed.

According to these results, the median processing speed of the model using VGG16, which is the lightest, exceeded 85 fps. The model using ResNet34, which showed the best results in the verification of object detection accuracy, was able to achieve 56.4 fps. The median is the benchmark. The reason for using the median as a benchmark here is that the minimum value is an outlier, as can be seen from the median and standard deviation.

Table IV shows the comparison with other methods. As compared to other models capable of 3D object detection, the proposed model is faster in terms of processing speed, partly because it consists of a simple network. YOLO3D, which is the fastest of the other models, is 40 fps using NVIDIA TITAN X, which is a GPU with the same performance as the one used in this study; therefore, it can be said that the proposed model greatly surpasses these models in terms of processing speed. One possible way to improve the accuracy while maintaining

TABLE IV. COMPARISON WITH OTHER METHODS

Model name	**GPU name**	**Speed [fps]**
YOLOv3	GTX 1080 Ti	74
Ours (ResNet34 + GCN II)	RTX 2080	56.4
YOLO3D	TITAN X	40
VoxelNet	RTX 2080	4.3
VoteNet	RTX 2080	3.2

TABLE V. RESULTS OF EVALUATION 3

Dataset Name	**Mean 3D IoU**
ImageNet	0.627
MS COCO	0.604
No Pre-train	0.584

the processing speed is to narrow the number of candidates for the bounding box by adding a few simple layers of GCN. This will also improve the accuracy of the estimation of the center point of the bounding box.

C. Evaluation 3: Effect of Pre-trained Model

Finally, we will verify the impact of the trained model of ResNet used as the backbone network by comparing the 3D IoU with a model trained on a different dataset. As described in Section II, the learned models used in Evaluations 1 and 2 were created using a dataset called ImageNet, which is designed for class classification. Therefore, a trained model using the MS COCO dataset [27] was prepared for comparison. For this validation, we used ResNet34 as the backbone network.

Table V shows the results of using each learned model. In this table, "No Pre-train" indicates the results when no trained model was used and random numbers were used as the initial values of parameters. According to this result, the best score was obtained with the trained model using ImageNet. It can also be seen that the score of the trained model is higher than that of the untrained model. In general, it is better to use trained models with object detection datasets such as MS COCO for object detection models like this one, so this result is contrary to that. In the proposed model, which uses 2D features for 3D object detection, the 3D features extracted by GCN are used to identify the 3D position. Therefore, it is possible that the classification results, which can be used directly for 3D, were more advantageous to the model than the 2D features. Furthermore, the proposed model uses unified detection as described in Section II, which means that it does not simply detect object positions but also classifies them.

From this, we found that the proposed model can obtain better results by using the classification task on the trained model, and that the features obtained from 2D are useful for 3D object detection.

IV. OBJECT DETECTION FOR REAL ENVIRONMENT

In order to check the robustness of the proposed model against occlusion, we conducted training and inference on real environmental data. As a dataset for this validation, we

Fig. 6 A scene where both the proposed model and YOLOv4 succeeded to detect people
Left: the output of YOLOv4; Right: the output of the proposed model

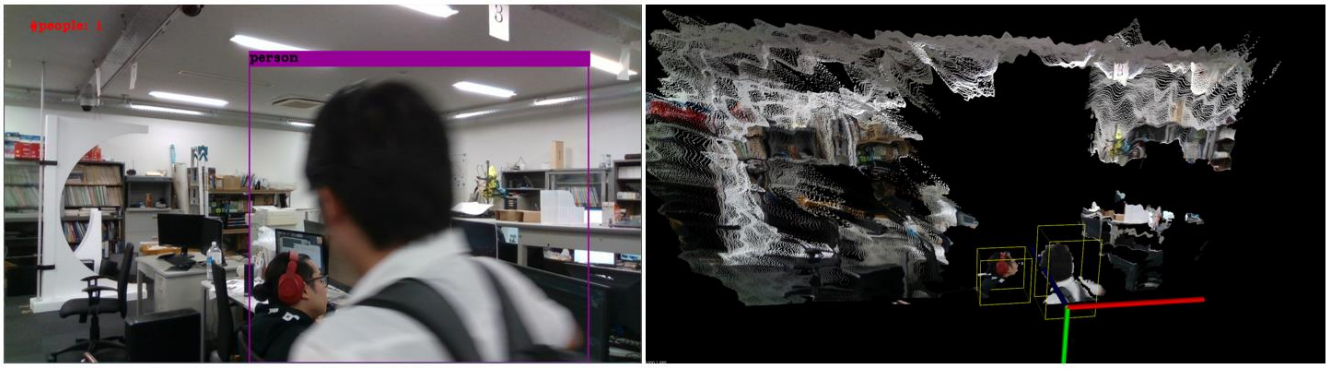

Fig. 7 A scene where the proposed model succeeded and YOLOv4 failed to detect people
Left: the output of YOLOv4; Right: the output of the proposed model

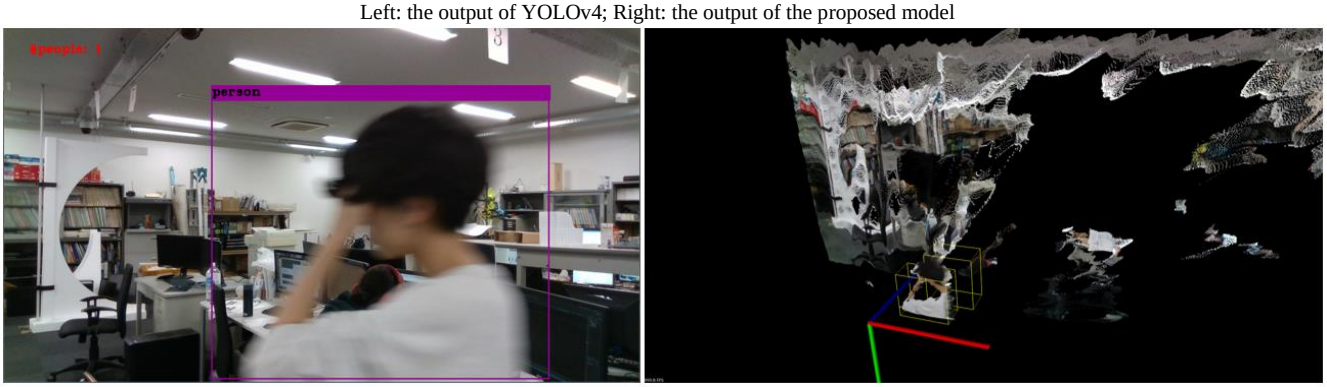

Fig. 8 A scene where both the proposed model and YOLOv4 failed to detect people
Left: the output of YOLOv4; Right: the output of the proposed model

collected RGB-D data of 25,506 indoor scenes in Chuo University. We used Intel RealSense D435 to acquire the data. This data was annotated with the correct label only for the human region. We trained the proposed model with this data and tried to detect people. We compared the results of this detection with those of YOLOv4. In particular, we analyzed the scenes in which occlusion occurred.

Fig. 6 shows a scene successfully detected people by both YOLOv4 and the proposed model. In this scene, we can see that YOLOv4 is able to handle occlusion. In addition to this, we can see that the proposed model predicts the depth and hidden parts of the scene. Fig. 7 shows a scene where YOLOv4 failed and the proposed model succeeded; YOLOv4 failed to detect the person behind, while the proposed model detected this person as well. Thus, the proposed model is robust against occlusion considering the depth information because it can detect a person who could not be detected by the 2D object detection. Fig. 8 shows a scene where both YOLOv4 and the proposed model failed to detect a person. The proposed method succeeds in outputting the bounding box, but fails in outputting the position. A possible reason for the failure of the 2D object detector is that the area of the person visible in the image was too small to be found. On the other hand, in the proposed model, although the target point cloud itself was obtained, the number of points was reduced due to the use of sparse point clouds, which made it difficult to obtain 3D features.

These results show that the proposed model is more robust against occlusion than the 2D object detector although it is a lightweight model.

V. Conclusion

We have presented a new object detection model using color and depth images. This achieved fast 3D object detection, which could not be done with existing models. Our method is a novel attempt to perform object detection in 3D space using color features and graph convolution. This idea has potential for application to 3D object detection by making simple extensions to the training model with color images.

In future work, the accuracy will be improved by further deepening and adapting to different scales.

References

[1] K. He, G. Gkioxari, P. Doll, and R. Girshick, "Mask R-CNN," in Proc. of the IEEE International Conference on Computer Vision (ICCV), pp. 2980–2988, 2017.

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 580–587, 2014.

[3] R. Girshick, "Fast R-CNN," in Proc. of the IEEE International Conference on Computer Vision (ICCV), pp. 1440–1448, 2015.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R CNN: Towards real-time object detection with region proposal networks," in Journal of IEEE Transaction on Pattern Analysis and Machine Intelligence (TPAMI), vol. 39, no. 6, pp. 1137–1149, 2015.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, real time object detection," in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779–788, 2016.

[6] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6517–6525, 2017.

[7] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," in arXiv preprint arXiv:1804.02767, 2018.

[8] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," in arXiv preprint arXiv:2004.10934, 2020.

[9] T. Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object selection," in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2117–2125, 2017.

[10] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 652–660, 2017.

[11] C. R. Qi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in Proc. of the Advances in Neural Information Processing Systems (NeurIPS), pp. 5099–5108, 2017.

[12] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep Hough voting for 3D object detection in point clouds," in Proc. of the IEEE International Conference on Computer Vision (ICCV), pp. 9277–9286, 2019.

[13] W. Ali, S. Abdelkarim, M. Zahran, M. Zidan, and A. E. Sallab, "YOLO3D: End-to-end real-time 3D oriented object bounding box detection from LiDAR point cloud," in Proc. of the European Conference on Computer Vision (ECCV) Workshops, 2018.

[14] M. Simon, S. Milz, K. Amende, and H. M. Gross, "Complex-YOLO: An euler-region-proposal for real-time 3d object detection on point clouds," in Proc. of the European Conference on Computer Vision (ECCV), pp. 197–209, 2018.

[15] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in Proc. of International Conference on Learning Representations (ICLR), 2016.

[16] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in Proc. of the International Conference on Machine Learning (ICML), pp. 1725–1735, 2020.

[17] S. Song, S. Lichtenberg, and J. Xiao, "Sun rgb-d: A rgb-d scene understanding benchmark suite," in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 567–576, 2015.

[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in arXiv preprint arXiv:1409.1556, 2014.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, 2016.

[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in Proc. of the European Conference on Computer Vision (ECCV), pp. 21–37, 2016.

[21] X. Liu and X. Di, "TanhExp: A smooth activation function with high convergence speed for lightweight neural networks," in arXiv preprint arXiv:2003.09855v2, 2020.

[22] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," in Proc. of the International Conference on Machine Learning (ICML) Deep Learning Workshop, 2015.

[23] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. F. Fei, "ImageNet: A large-scale hierarchical image database," in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 248–255, 2009.

[24] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in Proc. of the European Conference on Computer Vision (ECCV), pp. 746–760, 2012.

[25] A. Janoch, S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, and T. Darrell, "A category-level 3d object dataset: Putting the kinect to work," in Journal of Consumer Depth Cameras for Computer Vision, pp. 141–165, 2013.

[26] J. Xiao, A. Owens, and A. Torralba, "SUN3D: A database of big spaces reconstructed using sfm and object labels," in Proc. of the IEEE International Conference on Computer Vision (ICCV), pp. 1625–1632, 2013.

[27] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, "Microsoft coco: Common objects in context," in Proc. of the European Conference on Computer Vision (ECCV), pp. 740–755, 2014.