グラフ畳み込みを用いたカラー画像と深度画像による高速な 3 次元物体検出
Fast 3D Object Detection with RGB-D Images Using Graph Convolutional Network

精密工学専攻　２８号　高橋正裕
Masahiro Takahashi

1. 研究背景

物体を検出・認識することは，防犯やマーケティング等，様々な分野に応用が可能である．これらを高速かつ正確に行うことは，カメラを用いたシステムの能力全体を向上させることになるので，重要である．また，ステレオカメラを始めとした深度を計測可能なセンサが安価に手に入れることができるようになってきている．

近年，この分野においては，深層学習を 3 次元物体検出に応用する研究が多く行われている．その中でも PointNet[1]や PointNet++[2]は点群からの特徴抽出を可能とし，これらを物体検出や Semantic Segmentation タスクに応用した VoteNet[3]や YOLO3D[4]といったモデルが提案されている．しかし，これらの研究の多くは密な点群を扱うために計算コストが高く，リアルタイムでの運用が困難であり，高性能な GPU が必要となる．また，車載の LiDAR 等によって得られた広範囲の点群を用いることを前提にしているなど，大規模な環境を想定したものが多く，手軽に導入できるとは言えない．

そこで本研究では，RGB-D カメラから得られたカラー画像と深度画像を用いた軽量な 3 次元物体検出モデルを提案する．具体的には，カラー画像から特徴抽出を行ったのち，深度画像から疎な点群を作成することで，高速な物体検出を実現する．また，作成された点群に対してさらに GCN (Graph Convolutional Network)[5]を用い，奥行き情報を考慮した特徴抽出を行う．

本研究では，3 次元物体検出用データセットである SUN RGB-D データセット[6]を用いて提案モデルの物体検出精度を評価する．また，処理速度についても，カラー画像から特徴抽出を行う Backbone Network 別に比較し，評価を行う．結果として，提案モデルの物体検出精度は VoteNet 等の物体検出手法に及ばないものの，非常に高速かつ軽量なモデルにより 3 次元物体検出を実現することができた．

2. 提案手法

2.1 ネットワーク構造

本研究で提案するネットワーク構造を Fig. 1 に示す．ネットワークは大きく分けて，色特徴抽出部分(Backbone Network)，グラフ作成部分，3 次元特徴抽出部分の 3 つに分けられる．

色特徴抽出部分：既存の特徴抽出モデルである VGG16[7]や ResNet[8]を用い，カラー画像から特徴抽出を行う．これにより得られた高次元の特徴をそのまま以降のモデルで用いることで，クラス分類や物体検出に必要なカラー画像からの特徴を，3 次元物体検出に活用する．また，この部分は一般的なカラー画像から得られた特徴を用いることが目的であるため，物体検出器[9-13]と同様の学習済みモデルを用いることが可能である．

グラフ作成部分：まず，カラー画像から得られた高次元の特徴を属性値として付与した点群を，KNN(K-Nearest Neighbor)によって 16 近傍に対してエッジを持つグラフ構造に変換する．この時，特徴マップに合わせてスパースな点群を生成し，そこからランダムサンプリングを行う．これにより，以降のモデルに入力する点の数を固定するとともに，ランダムサンプリングを行うことで Data Augumentation と同様の効果を得ることができると考えられる．

3 次元特徴抽出部分：グラフ構造となった点群を，6 層の GCN (Graph Convolutional Network)により，近傍点の位置を考慮した，特徴抽出やバウンディングボックスの推定を行う．今回のモデルは点群の座標を扱うため，入力に負の値が存在する．そのため，活性化関数には，LeakyReLU[14]と同様に負の値を扱うことができる TanhExp[15]を用いる．出力形式は YOLO (You Only Look Once)[12]や SSD (Single Shot Detection)[13]に共通する手法である Unified Detection を元に作成した．Unified Detection とは，バウンディングボックス

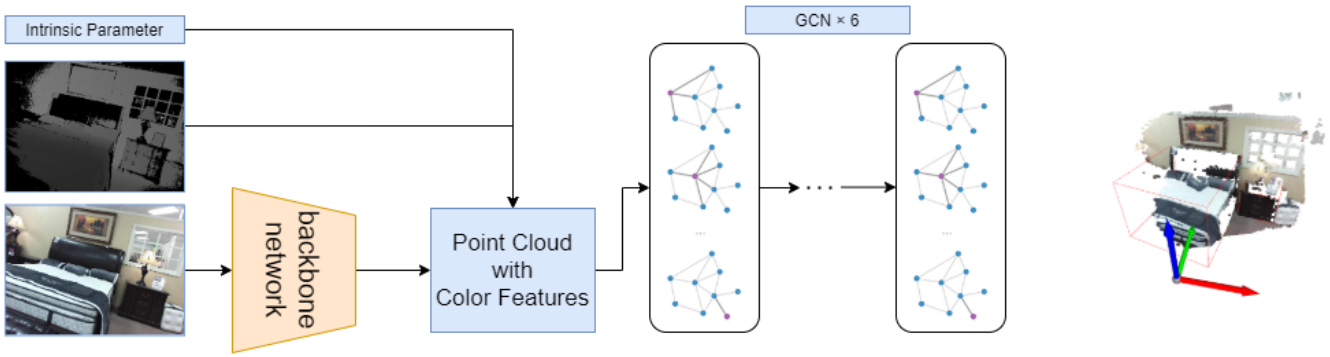

Fig. 1　Network Architecture of the Proposed Model
Input: Color, Depth Image, and Intrinsic ParameterOutput: 3D Bounding Boxes and Classification

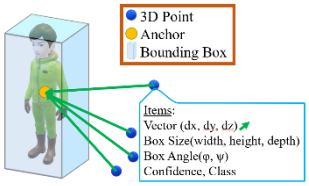

Fig. 2　Output of Proposed Model

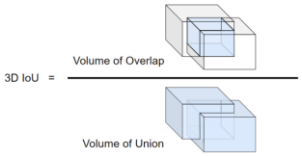

Fig. 3　Description of 3D IoU

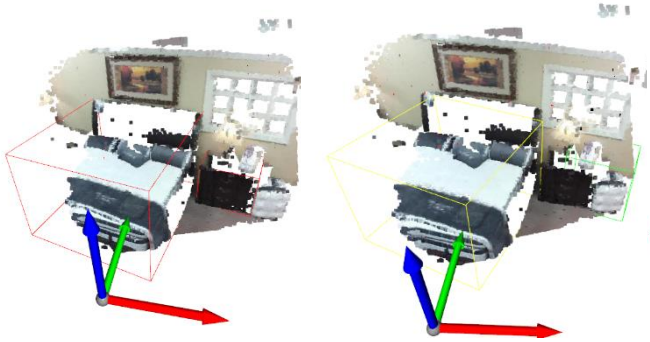

Fig. 4　Ground Truth and Model Output of Successful Scene

Right: Ground Truth, Left: Model Output

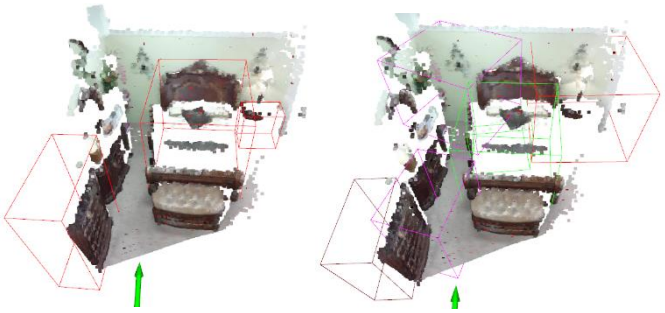

Fig. 5　Ground Truth and Model Output of Failure Scene

Right: Ground Truth, Left: Model Output

の座標やクラス分類結果等を 1 つのテンソルに格納することで，クラス分類と領域特定を同時に出力することが可能となる手法である．Fig. 2 のように，提案モデルでは，出力の各 3 次元座標が，注目点からバウンディングボックスの中心点までのベクトル (dx, dy, dz)，バウンディングボックスの大きさ(width, height, depth)，バウンディングボックスの角度(φ, ψ)，出力ベクトルの信頼度，クラス分類結果を出力する．出力サイズは，バッチサイズ×3 次元点の数×(9+クラス数)となる．最終的に，これらのバウンディングボックス情報とクラス分類情報が 1 つのテンソルとしてまとめて出力される．そのため，バウンディングボックスの位置特定とクラス分類を別々に行う場合と比べて，モデルに大きな分岐がなくコンパクトになるので，高速な物体検出が可能となる．

以上のネットワーク構造により，提案モデルはカラー画像と深度画像から 3 次元バウンディングボックスを出力する．

2.2　学習と推論

提案モデルでは，Backbone Network に用いる VGG や ResNet は，ImageNet[16]によって事前学習を行ったものを用いる．これにより，画像中で事前に特徴のある領域がある程度提示された状態で 3 次元物体の推定を行うことができる．

提案モデルの出力形式は Unified Detection であるため，誤差関数も YOLO に類似したものを学習に用いる．この誤差関数は，バウンディングボックスの中心座標へ向かうベクトルの MSE (Mean Squared Error)，バウンディングボックスの大きさの MSE，バウンディングボックスの回転角度の MSE，ベクトルの信頼度の BCE (Binary Cross Entropy)誤差の合計で表される．誤差関数は式(1)のようになる．

$$
\begin{aligned}
&Loss = \\
&\lambda_{bb}\{MSE((dx,dy,dz)_{out},(dx,dy,dz)_{tar}) \\
&+ MSE((\varphi,\psi)_{out},(\varphi,\psi)_{tar}) \\
&+ BCE(cos\theta_{out},cos\theta_{tar}) \\
&+ BCE(cls_{out},cls_{tar})\} \\
&+ \lambda_{nobb}\{BCE(cos\theta_{out},cos\theta_{tar})\}
\end{aligned}
\tag{1}
$$

ここで，下付き文字 out はモデルの出力，tar は教師データを表している．cosθ は，バウンディングボックスの中心点へ向かうベクトルと正解のベクトルの角度 θ から得られる．また，λ_{bb} と λ_{nobb} は，それぞれ物体が存在する，もしくはしないときに計算される誤差に対する係数で，今回は λ_{bb}=1，λ_{nobb}=10 を用いる．ここで λ_{nobb} を大きめに設定する理由としては，ここを小さな値に設定すると，誤検出を抑制するための誤差が小さくなり，結果として誤検出が多発してしまうためである．cls はクラス分類結果であり，one-hot ベクトルで出力されるため，BCE によって誤差計算を行う．

得られた候補から最適なバウンディングボックスを選択する代表的な手法としては，R-CNN に用いられている NMS (Non Maximum Suppression) がある．これは IoU (Intersection over Union)と呼ばれる領域の重なり度合いを表すスコアをもとに，同じ物体に対して推定されたバウンディングボックスを消去する方法である．YOLO3D 等の 3 次元情報を扱う物体検出器は，センサ正面方向と垂直方向の 2 方向から 2 次元に対する IoU を計算し，NMS によるバウンディングボックスの選択を行っている．しかし，この方法では同じ 3 次元のバウンディングボックスに対して 2 回 IoU を計算していることとなり，GPU による並列計算を行う上で非効率的である．そこで，提案モデルでは，Fig. 3 のような体積の重なり度合いを表す 3DIoU を定義し，これをもとに NMS を実行する．これにより，IoU 計算は GPU 上で 1 回しか行わず，処理速度の改善が期待できる．NMS における IoU のしきい値は，YOLO 等においては 0.5 が用いられていたが，提案モデルでは最も良いスコアであった 0.6 を用いた．

3.　提案モデルの性能検証

提案モデルの有効性を確認するため，SUN RGB-D データセットを用いて精度を検証した．また，リアルタイム性についても検証するため，Backbone Network を変えたときの処理時間の比較を行った．検証に用いたマシンのスペックは，

Table 1　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

CPU が Intel Core i7 8770K，GPU が NVIDIA RTX2080 であった．

3.1 検証 1：物体検出精度検証

物体検出精度の検証として，SUN RGB-D データセットのうち， bed, table, sofa, chair, toilet, desk, dresser, night stand, bookshelf, bathtub の計 10 クラスを 100 エポック分学習させた．バッチサイズは 3，入力画像サイズは 224×224[pixel], 学習係数は 0.0001 に設定し，最適化手法には Adam (Adaptive moment estimation)を用いた．

物体検出の Ground Truth と提案モデルによる物体検出結果の成功例を Fig. 4 に示す．このシーンでは，bed と night stand が正解として与えられているが，どちらもかなり正確に検出できているといえる．特に bed に関しては角度やサイズも正確に検出できており，大きい物体に関しても検出が可能であることがわかる．night stand に関しては物体の中心推定に誤差が生じているが，サイズを正しく推定できていることがわかる．続いて，失敗例を Fig. 5 に示す．このシーンでは，それぞれの物体は検出できているものの，各物体の中間に誤検出が発生していることがわかる．これら 2 つの例を比較すると，点群の質に差があることがわかる．成功例の点群は各物体に対して得られている点群にノイズが少なく，点群がまとまっており，その結果近傍点を考慮した GCN による特徴抽出がうまくいったと言える．一方，失敗例においては，物体毎に得られている点群にまとまりがなく，物体が存在しない部分に関しても点群のばらつきが多いため，近傍点探索によるグラフの作成がうまくいかなかったと考えられる．

Backbone Network を変えたときの 3D IoU による結果の平均値を Table 1 に示す．この結果によると，ResNet34 を Backbone として用いたときの結果が最も良いことがわかる．ResNet34 より深い層を持つモデルでうまくいかなかった理由としては，今回用いた入力画像のサイズが小さく，出力サイズが足りなかったことが考えられる．3D IoU のスコアとしては，0.5 を超えているため，物体をある程度正確に検出ができているといえる．しかし同じ 3 次元物体検出用モデルである VoteNet のスコアの 0.83 と比較すると，精度面では及ばなかったことがわかる．この理由としては，VoteNet では密な点群を用いて検出しているのに対し，提案モデルではスパースな点群を用いた検出であるため，その分 3 次元推定が困難になっていると考えられる．しかし，Fig. 5 のように誤検出が多いものの物体位置の特定はできているため，学習エポック数の増加や学習係数の調整次第では解決可能であると考えられる．

Table 2　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

3.2 検証 2：処理速度検証

続いて，提案モデルの処理速度の検証を行った．検証方法としては，合計 200 フレームを推論させ，その結果を用いて統計量を算出した．

Table 2 は，Backbone Network を変えたときの処理速度の結果である．この結果によると，最も軽量な VGG16 を用いたモデルは，処理速度の中央値が 85[fps]を超えており，物体検出精度検証において一番良い結果であった ResNet34 を用いたモデルも，56.4[fps]を出せているため，十分にリアルタイム性を保持できていると言える．ここで，中央値をベンチマークとした理由としては，中央値や標準偏差からわかるように，最小値が外れ値を取っているためである．

3 次元物体検出を行うことが可能な他のモデルと比較して，処理速度に関しては，提案モデルがシンプルなネットワークで構成されていることもあり，高速な検出を実現することができた．他のモデルで最も高速な YOLO3D も，今回用いたものと同等の性能を持つ GPU である NVIDIA TITAN X を用いて 40[fps]であるため，処理速度に関してはこれらを大きく上回ったといえる．処理速度を維持したまま精度向上を行う方法としては，シンプルな数層の GCN を追加することでバウンディングボックスの候補数を絞ることが考えられる．また，これによりバウンディングボックス中心点の推定精度の向上も見込める．

3.3 検証 3：学習済みモデルによる影響の検証

最後に，Backbone Network として用いていた ResNet の学習済みモデルを，別なデータセットで学習させたモデルを用いた状態で 3D IoU を比較することで，学習済みモデルが与える影響について検証を行う．検証 1 と検証 2 において用いていた学習モデルは，2.2 節で述べた通り，ImageNet と呼ばれるクラス分類を目的としたデータセットを用いて作成していた．そこで，比較対象として MS COCO[17]というデータセットを用いた学習済みモデルを用意した．今回の検証には，ResNet34 を Backbone Network として用いた．

Table 3 に，各学習済みモデルを用いた結果を示す．この表における No Pretrain は，学習済みモデルを用いず，パラメータの初期値に乱数を用いた時の結果を示している．この結果によると，ImageNet を用いた学習済みモデルが最も良いスコアとなった．また，学習済みモデルを用いたほうが，用いないものに比べてスコアが上がっていることがわかる．また，

Table 3　Results of Experiment 3

Dataset Name	Mean 3D IoU
ImageNet	0.627
MS COCO	0.604
No Pretrain	0.584

一般的には，今回のような物体検出を目的としたモデルに対しては，MS COCO のように物体検出を目的としたデータセットを用いた学習済みモデルを用いるほうが良いとされているため，この結果はそれに反するものとなった．3 次元の物体検出に対して 2 次元の特徴を利用する提案モデルでは，3 次元位置の特定には GCN により抽出される 3 次元の特徴が用いられる．そのため，2 次元に対する物体位置の特徴よりも，そのまま 3 次元に用いることができるクラス分類結果のほうがよりこのモデルにとって有利に働いたということが考えられる．さらに，提案モデルでは，2.1 節で述べた通り，Unified Detection を用いている．これにより，単純に物体位置を検出するのではなく，クラス分類も同時に行うというタスクが GCN 上で行われているため，よりクラス分類結果が重視されたのではないかと考えられる．

以上より，提案モデルでは，学習済みモデルにクラス分類タスクを用いることで，より良い結果を得られること，ならびに 2 次元から得られた特徴が 3 次元物体検出に有益であることが分かった．

4. 結言

本研究では，RGB-D から 3 次元のバウンディングボックスを出力可能で軽量なモデルを作成した．提案モデルでは，カラー画像から得た特徴を点群とともに GCN へ入力することで，バウンディングボックスの奥行き方向の位置と長さを取得した．

今後の展望としては，更なる深層化や異なるスケールへの対応を行うことで，精度向上を図る．

参考文献

(1) 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.

(2) 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.

(3) 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. 9977-9286, 2019.

(4) 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.

(5) T. N. Kipf and M. Welling, “Semi-Supervised Classification with Graph Convolutional Networks,” in Proc. of International conference on Learning Representations (ICLR), 2016.

(6) 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.

(7) K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large Scale Image Recognition,” in arXiv preprint arXiv:1409.1556, 2014.

(8) 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), 2016.

(9) 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.

(10) S. Ren, K. He, R. Girshick, and J. Sun, “Faster R CNN: Towards real time object detection with region proposal networks,” in Proc. of Conference on Neural Information Processing Systems (NeurIPS), 2015.

(11) 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.

(12) J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” in arXiv preprint arXiv:1804.02767, 2018.

(13) 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.

(14) A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in Proc. of International Conference on Machine Learning (ICML), p. 3, 2013.

(15) 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.

(16) 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.

(17) T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in Proc. of the European Conference on Computer Vision (ECCV), pp. 740-755, 2014.