TH-NET: A METHOD OF SINGLE 3D OBJECT TRACKING BASED ON TRANSFORMERS AND HAUSDORFF DISTANCE

Zihao Zhang *Nan Sang* *Xupeng Wang**

University of Electronic Science and Technology of China
School of Information and Software Engineering
No.4, Section 2, North Jianshe Road

ABSTRACT

3D object tracking is the key of automatic driving. We propose a new 3D object tracking method called Transformer-Hausdorff Net (TH-Net). It contains three main modules: Feature Extraction, Feature Fusion, and Proposal Generation. The Feature Extraction module extracts features from the template and search area, where the permutation-invariable Transformers is leveraged to deal with the point cloud's disorder and sparsity. The features of template and search area are then fused by the Feature Fusion module to generate the tracking clues. Based on the tracking clues, we further generate 3D target proposal and execute verification in Proposal Generation module. TH-Net achieves a performance improvement in contrast to the state-of-the-art work on KITTI and NuScenes dataset.

***Index Terms*—** single object tracking, Transformers, Hausdroff Distance, point cloud, proposal

1. INTRODUCTION

3D object tracking is vital to autonomous driving and augmented reality. However, sparsity and disorder of point cloud bring difficulties for 3D object tracking. As a result, some 3D tracking works[1, 2, 3, 4, 5] are based on mature 2D approaches instead of 3D point cloud. But 2D data is limited by occlusion and variable light intensity. These factors also influence tracking performance based on 2D data. To address the sparsity and disorder of point cloud, PointNet, DeepSets[6, 7] uses symmetric function to learn the feature of point cloud and achieve outstanding performance in classification and segmentation tasks.

The pioneering work of 3D object tracking using raw point cloud is SC3D[8]. SC3D completes the template matching in the search area by point cloud encoder[9] and shape completion network. But the shape completion Network leads to considerable memory and time overhead. Besides, Candidate shapes[8], used in SC3D, degrades the generalization performance of the model. Referring to the structural ideas of SC3D, P2B[10] is proposed. P2B utilizes the cosine distance to calculate the point-by-point correlation between the template and the search area. Then it generates tracking proposals using the RPN network[11]. P2B has some defects: 1) Cosine similarity is easily affected by deviation points, and it ignores the relationship between each point and the whole point cloud. 2) The representation ability of PointNet Siamese network is limited due to the defect of symmetric function[12]. 3D-SiamRPN[13] follows the P2B structure and use PCW-Xcorr[13] to tracking target, but ignores fine geometric information.

In order to solve the shortcomings of previous work, we propose a 3D tracking structure called TH-Net which can be trained end-to-end. We first use permutation-invariable Transformers structure to extract features from template and search area, which can avoid the shortcomings of symmetric functions[12]. Based on the extracted features, we leverage Hausdroff Distance to calculate the correlation between template and search area for template matching in the search area. Hausdroff Distance takes into account the correlation between each point and the whole point cloud. Last but not least, we implement the proposal generation and verification based on idea of[14], which divide the proposal generation into location prediction and orientation prediction. Generally, TH-Net is exemplified in Fig. 1.

Experimental results on the KITTI[15] and NuScenes[16] show that TH-Net outperforms the state-of-the-art methods[8, 10, 13]. On KITTI dataset, TH-Net surpasses previous works on average Success and Precision[17] (4% on Success and 6% on Precision). TH-Net runs in real-time at 27.2 FPS on a single NVIDIA 1080Ti GPU.

In summary, the main contributions are listed as follows:

First, we proposed a 3D object tracking method only using point cloud, which can be trained end-to-end.

Second, we propose a new method of tracking clues generation based on the Hausdorff distance, which can generate informative tracking clues for 3D object tracking.

Third, we utilize Siamese Network with Transformers as backbone to extract point cloud features, which can handle the point cloud's disorder and sparsity.

Fourth, Experiments show that our method achieves a superior performance and can run in real-time.

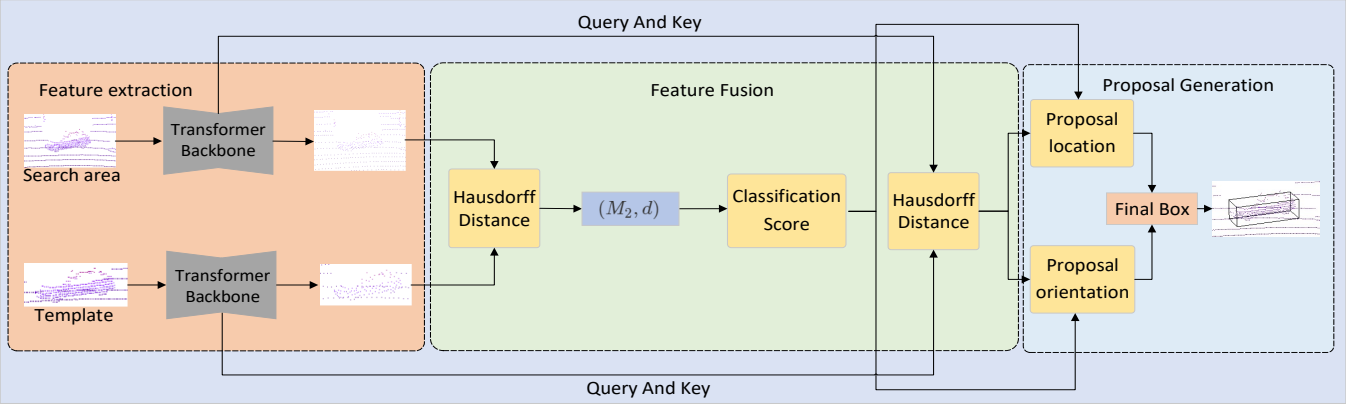

Fig. 1. **An overview structure of TH-Net.** TH-Net contains three modules: feature extraction, feature fusion, and proposal generation. Feature extraction module utilizes Transformers Backbone to extract feature. Feature fusion leverages the Hausdorff Distance to generate tracking clues. In the module of proposal generation, center point and surface point are respectively used for orientation and location predictions of proposal.

2. METHODS

TH-Net receives a search area point cloud and a template point cloud as input. Template is the target that needs to be tracked. TH-Net looks for the target that matches the template in the search area. Finally, TH-Net generates target proposal to "segment out" the matched target in search area as the tracking results. The whole process is shown in Fig. 1. TH-Net can be divided into three parts: Feature Extraction, Feature Fusion and Proposal Generation.

2.1. Feature Extraction

TH-Net extracts template and search area feature by the Transformers Siamese Network. We represent the template and search area in the same form $S = \{x_i\}_{i=1}^{N}$, because template and search area share the weight of Siamese network. x_i means the 3D location of a point in point cloud(S). N means the number of points in point cloud. Input is represented as $S = \{x_i\}_{i=1}^{N}$. In output, each point can be expressed as $[x_i; f_b] \in R^{3+d_1}$ where f_b means the feature of a point.

In Transformers Siamese Network, we first use the SA layer[6] to down-sample point cloud. Then we get down-sampled(seed) points $S_{sa} = \{x_i, f_{sa}\}_{i=1}^{M}$. f_{sa} means the feature of a down-sampled(seed) point.

Then, we leverage Transformers to further extract feature. We designed the permutation-invariable Transformers which can handle point cloud's sparsity and disorder. This Transformers is called offset-Transformers. We formalize it as follows:

$$q_i = f_{sa}W_q, k_i = f_{sa}W_k, v_i = f_{sa}W_v \tag{1}$$

$$attention = softmax(mlp(q_i - k_i)) \tag{2}$$

$$f_b = mlp(f_{sa} - mlp(attention(dot)v_i)) \tag{3}$$

We modified the traditional Transformers[18] to use the offset ((2) and (3)) to generate attention for fine local geometric information. q_i, k_i, v_i are query, key, value of Transformers[18]. W means the weight parameters of Transformers. $mlp()$ denotes linear layer implemented with $1 * 1conv$. $softmax$ means the softmax layer in [18]. dot means vector dot product. This structure is permutation-invariable and can handle the disorder and sparsity of the point cloud. We use three Transformers continuously here.

2.2. Feature Fusion

In the Feature Fusion module, TH-Net generate the tracking clues by Hausdorff Distance.

$S_t = \{x_i + f_t\}_{i=1}^{M_1}$ denotes Template feature and $S_s = \{x_i + f_s\}_{i=1}^{M_2}$ denotes search area feature.

$$offset_{i,j} = |f_t - f_s|, f_t \in S_t, f_s \in S_s \tag{4}$$

$offset()$ calculates offsets point by point. The following steps are as shown in Fig. 2. We apply Max pool and min pool on $offset$ respectively and then combine them to obtain Hausdorff Distance $(M_1, M_2, 2)$. We enhance features by attaching template feature to Hausdorff Distance $(M_1, M_2, 2 + f_t)$. Then $MLP(1 * 1conv)$ further fuses points' feature. Finally, we generate the fusion feature (M_2, d) by Max pool.

Classification score. As shown in Fig. 1, we use the Hausdorff Distance between template and search area to learn MLP for points' classification score. Points locate on the ground-truth object are positive. And points of other circumstances are negative. We adopt the binary cross-entropy in [10] to generate the loss L_{cla}.

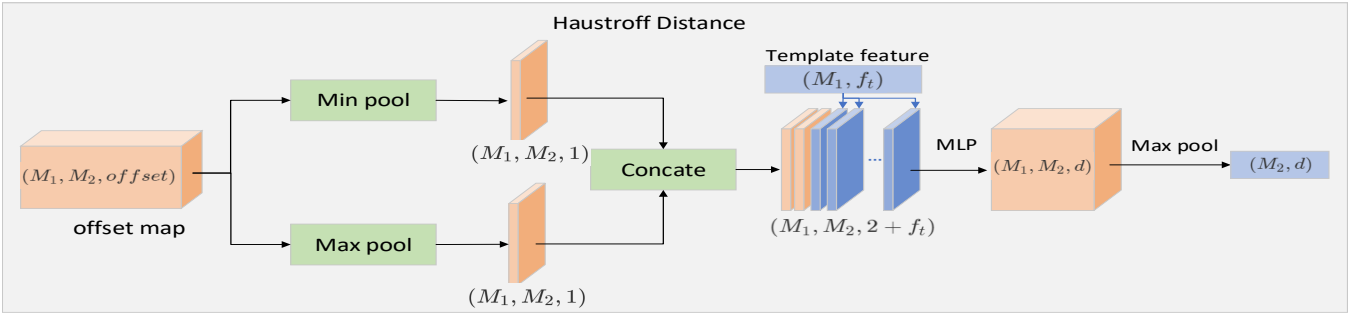

Fig. 2. **Hausdorff Distance.** We computes the distance between each point, and then combines the maximum and minimum distance to get Hausdorff Distance.

2.3. Proposal Generation

Proposal generation is divided into two parts. Location of proposal is predicted by clusters of central points. Orientation of proposal is predicted by clusters of surface points. The reason for this is that center points lose geometric info but remain precise location knowledge. And surface points possess geometric knowledge without precise location knowledge[14].

We concatenate the q and k of each Transformers (three layers in total) in (1). Then the symmetric function (Max pool is adopted here) is applied to get $query$ and key in (5)(6).

As the Fig. 1 shows, $query_t, key_t, query_s, key_s$ represents key and $query$ in (5). Subscript t means the feature of template, and s means the features of search area. The formulas are as follows:

$$(M_2, f_1) = Hausdorff(query_t, query_s) \tag{5}$$

$$(M_2, f_2) = Hausdorff(key_t, key_s) \tag{6}$$

We generate center points by Hough voting[11]. Inspired by [14], surface points are clustered by center points clusters' index. Location and orientation of the proposal are predicted respectively by center points' clusters and surface points' clusters.

$$\begin{aligned}(M_2, \Delta f_1) &= HoughVoting((M_2, f_1)) \\ (M_2, f_{center}) &= (M_2, f_1) + (M_2, \Delta f_1)\end{aligned} \tag{7}$$

$$(C, location), index = mlp(cluster(M_2, f_{center})) \tag{8}$$

Hough Voting[11] is implemented by (7) where (M_2, f_{center}) means predicted center points. We cluster center points to predict proposal's location. $cluster()$ is implemented by PointNet[6]. C means the number of proposals.

$$(C, orientation, score) = mlp(cluster((M, f_2), index)) \tag{9}$$

$index$, clustering index of center points, serves as an index to cluster surface points in (9). $score$ denotes the confidence score of each proposal[11]. $orientation$ is the orientation of proposal. After the above proposal generation, we followed the P2B[10] strategy for proposal verification.

Proposal loss $L_{proposal}$ is designed following previous works[8, 10, 13]. Specially, the proposals' centers within 0.3 meters from the object centers are positive, and proposals' centers more than 0.6 meters away are negative. Others are not included in the calculation.

2.4. loss function

We adopt L1 loss for the final box(verified proposal), cross-entropy loss for classification and proposal generation. L_{reg} is utilized to regress center points of (7) following the setting of [11]:

$$L_{reg} = \frac{1}{M}||\Delta f_i - \Delta f_{gt}||\delta[point_i \; on \; ground truth] \tag{10}$$

Δf_i denotes the voting offset in Eq. 7. Δf_{gt} denotes the ground-truth offset of coordinate. δ denotes points locating on the surface of the ground-truth object. M denotes the number of points. The overall loss is as follow, in which $\alpha = 0.2$, $\beta = 1.5$ and $\gamma = 0.2$.

$$L = L_{reg} + \alpha L_{cla} + \beta L_{proposal} + \gamma L_{finalbox} \tag{11}$$

3. EXPERIMENTS

3.1. Datasets

We adopted the KITTI[15] tracking data as a benchmark. KITTI[15] includes 21 outdoor scenes and 8 types of objects. We test TH-Net with the main types: vehicle and person. We divided the input video into 20 frames: 0-16 frames as the training set, 17-18 as the validation set, and 19-20 as the test set. We also used the NuScenes[16] dataset which contains traffic point cloud in dense scenes. We adopt NuScenes

	method	Car	Pedestrain	Van	Cyclist	Mean
		6424	6088	1248	308	14068
Success	SC3D	41.3	18.2	40.4	**41.5**	31.2
	P2B	56.2	28.7	40.8	32.1	42.4
	3D-SiamRPN	**58.22**	35.23	45.67	36.61	46.67
	Ours	57.78	**41.58**	**53.6**	40.01	**50.17**
Precision	SC3D	57.9	37.8	47	**70.4**	48.5
	P2B	72.8	49.6	48.4	44.7	60
	3D-SiamRPN	**76.24**	56.22	52.58	49.03	64.91
	Ours	73.88	**69.62**	**66.5**	50.74	**70.88**

Table 1. **3D single object tracking results on KITTI**

	method	Car	Truck	Trailer	Bus	Mean
		32302	8646	2297	2215	45460
Success	P2B	33.28	24.20	28.90	26.83	31.01
	3D-SiamRPN	**35.96**	25.14	**30.01**	27.41	33.18
	Ours	35.28	**27.73**	29.82	**28.09**	**33.22**
Precision	P2B	34.88	27.54	27.56	25.26	32.64
	3D-SiamRPN	36.81	**28.05**	**28.76**	25.50	34.18
	Ours	**37.08**	27.94	26.39	**27.25**	**34.32**

Table 2. **3D single object tracking results on NuScenes**

default dataset partition to generate training set and test set. Tracklets are created following [8, 10].

We evaluated our Transformers backbone on the ModelNet40 [19] following the sampling and training strategy of PCT [20] and PointNet [6].

3.2. Implementation details

Template and search area. We randomly sampled 512 points for template and sampled 1024 points for search area.

Siamese network. We down-sampled the template to 64 and search area to 128 points by two SA-layers[6] with the radius of 0.3 and 0.5. q, k, v in offset-Transformers were all 256 dimensions.

Training. We trained TH-Net for 100 iterations from scratch with the same data augmentation strategy following [10]. Adam[21] was used here with learning rate decaying 0.2 for every 10 epochs.

Test. The initial template used the ground truth point cloud, and then used the tracking result of the previous frame as the template.

Metric. We followed the setting of [17]. $Success$ is Intersection over Union(IOU) between final box(in Fig. 1) and ground truth box. $Precision$ is calculated by Area Under the Curve(AUC).

3.3. Comparing with previous methods

We compared TH-Net with existing single object tracking methods[8, 10, 13]. The results are given in Table 1 and Table 2. TH-Net outperforms previous works by 4% Success and 6% Precision evenly on KITTI . And TH-Net also performs well on the NuScenes dataset.

For the deformable pedestrian in KITTI[15], TH-Net achieves state-of-the-art results. Point clouds in NuScenes[16] are dense and complex, and TH-Net has less improvement in NuScenes compared to previous work. We believe that the TH-Net is difficult to handle dense scenes.

method	accuracy
PointTransformers[22]	89.3
PointTransformers[23]	92.4
PCT[10]	92.5
Centroid Trasnformers[24]	92.3
Our default setting	**92.8**

Table 3. **Transformers Backbone result.** The comparison accuracy is in ModelNet40[19]

method	car		pedestrain	
	Success	Precision	Success	Precision
Cosine distance	55.32	72.13	36.94	58.99
PCW-Xcorr[13]	54.39	72.75	38.17	65.26
PW-Xcorr[13]	55.55	71.39	40.49	68.34
Hausdorff	**57.78**	**73.88**	**41.58**	**69.62**

Table 4. **Feature fusion comparison.**

3.4. Ablation study

Transformers Backbone: Besides the Transformers backbone in TH-Net, there are some works about 3D point cloud Transformers backbone[23, 22, 20, 24]. We evaluated them all on ModelNet40[19] with Adam[21]. The learning rate decayed 0.8 for every 50 epochs. We trained each model for 5000 iterations from scratch. The results are shown in Table 3. Our offset-Transformers achieves the best result compared with these methods.

Feature fusion: We test other methods of feature fusion by replacing Hausdroff Distance in Fig. 1, as shown in Table 4. We compared the Hausdorff distance (tracking clues generation) in Fig. 1 with cosine distance[10, 8], PCW-Xcorr[13] and PW-Xcorr[13]. The results indicate Hausdorff Distance's superior performance in feature fusion.

3.5. Running Speed

TH-Net spends 37.1 ms for forwarding propagation and achieves 27.2 FPS on a single NVIDIA 1080Ti GPU, which indicates that TH-Net achieves real-time.

4. CONCLUSIONS

In this paper, we propose an object tracking network named TH-Net. It extracts features by Transformers Siamese network. It then fuses features via Hausdorff Distance to get tracking clues. Finally, the proposal task is decomposed by characteristics of points' location. TH-Net outperforms previous works in experiments, which proves its superiority in 3D object tracking. In the future, we would seek more effective feature fusion methods and try to extend TH-Net to multi-object detection.

5. REFERENCES

[1] Alireza Asvadi, Pedro Girão, Paulo Peixoto, and Urbano Nunes, "3d object tracking using rgb and lidar data," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 1255–1260.

[2] Adel Bibi, Tianzhu Zhang, and Bernard Ghanem, "3d part-based sparse tracker with automatic synchronization and registration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[3] Ugur Kart, Alan Lukezic, Matej Kristan, Joni-Kristian Kamarainen, and Jiri Matas, "Object tracking by reconstruction with view-specific discriminative correlation filters," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[4] Ugur Kart, Joni-Kristian Kamarainen, and Jiri Matas, "How to make an rgbd tracker ?," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[5] Ye Liu, Xiao-Yuan Jing, Jianhui Nie, Hao Gao, Jun Liu, and Guo-Ping Jiang, "Context-aware three-dimensional mean-shift with occlusion handling for robust object tracking in rgb-d videos," *IEEE Transactions on Multimedia*, vol. 21, no. 3, pp. 664–677, 2019.

[6] C. R. Qi, Y. Li, S. Hao, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," 2017.

[7] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola, "Deep sets," 2017.

[8] S. Giancola, J. Zarzar, and B. Ghanem, "Leveraging shape completion for 3d siamese tracking," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[9] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv preprint arXiv:1706.02413*, 2017.

[10] Haozhe Qi, Chen Feng, Zhiguo Cao, Feng Zhao, and Yang Xiao, "P2b: Point-to-box network for 3d object tracking in point clouds," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 6328–6337, 2020.

[11] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3d object detection in point clouds," *IEEE*, 2019.

[12] Christian Bueno and Alan G. Hylton, "Limitations for learning from point clouds," 2020.

[13] Zheng Fang, Sifan Zhou, Yubo Cui, and Sebastian Scherer, "3d-siamrpn: An end-to-end learning method for real-time 3d single object tracking using raw point cloud," *IEEE Sensors Journal*, vol. 21, no. 4, pp. 4995–5011, 2021.

[14] Yanxian Chen, Huimin Ma, Xi Li, and Xiong Luo, "S-votenet: Deep hough voting with spherical proposal for 3d object detection," in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 5161–5167.

[15] Andreas Geiger, Philip Lenz, and Raquel Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[16] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11621–11631.

[17] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang, "Online object tracking: A benchmark," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," 2017.

[19] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[20] Liu ZN. et al Guo MH, Cai JX, "Pct: Point cloud transformer," 2021.

[21] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," 2014.

[22] Nico Engel, Vasileios Belagiannis, and Klaus Dietmayer, "Point transformer," 2020.

[23] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun, "Point transformer," 2020.

[24] Lemeng Wu, Xingchao Liu, and Qiang Liu, "Centroid transformers: Learning to abstract with attention," 2021.