Transformer-based 3D Single Object Tracker

Akshay Kumar Sureddy
akshaykumars@umass.edu
University of Massachusetts Amherst

Roshini Pulishetty
rpulishetty@umass.edu
University of Massachusetts Amherst

Abstract

Recently, Transformers have revolutionized the field of natural language processing and visual analysis by robustly modelling the dependencies regardless of their distances. While they have been largely explored for sequence prediction and object detection, their potential for fusing features of different entities remains under-explored. We present this idea of leveraging transformers capability in 3D object tracking to embed template features into the search area in order to propose target regions. Thus, we propose an effective tracker, that integrates transformers within the state-of-the-art Point to Box tracker, termed P2B. Following the classic Siamese-based paradigm, the tracker contains 3 modules - feature extraction, feature matching and joint target proposal and verification. It initially extracts the features from the template and the search area respectively. Consequently, the transformer augments the search area seeds with the template seeds, which serves as an appropriate input for the regressing the potential target centers via Hough voting, which is further, strengthened by ensemble power and targetness scores. Experiments on KITTI Velodyne dataset reveal that though our tracker is in-par with P2B, that is, within $\pm 10\%$ on all categories, it outperforms P2B and LTTR baselines on non-rigid objects.

1. Introduction

Tracking objects has been instrumental in many scenarios, like autonomous driving, robotics, and surveillance system. While 2D *Visual Object Tracking* requires analysing the images in sequence, these images rely heavily on RGB information, which degrades their performance in dark or illumination-changing environments. Yet, some 3D object trackers deploy RGB-D sensors to capture the scenes, which inherit the same characteristics and challenges of 2D trackers. However, LIDAR sensors are more widely deployed and are particularly useful since they are not only less sensitive to the light and weather variations but also sense geometry accurately for a large spectrum of driving conditions.

In a sequence of point clouds, the objective is to track the location and orientation of an object in the search area given the template point cloud. Nevertheless, using only point clouds for 3D SOT poses a spectrum of challenges. (1) Owing to the sparsity and incompleteness of the point clouds, localizing not only non-rigid objects but also rigid objects is difficult. (2) They are disordered, demanding the permutation in-variance of the network. (3) Tracking an object in 3D requires output in higher dimension $(x, y, z, w, h, l, \theta)$ thus, higher computational complexity. Thus, this makes the task complicated and demands modules for each subproblem, rather than a simple conventional CNN.

The first pioneer effort on this problem statement is SC3D [13]. This work leveraged shape completion to regularize feature learning on point set while using Kalman filters for template matching. Nonetheless, this network is not end-to-end trainable and shape completion brings strong class prior, reducing generalizability. However, P2B tracking overcomes these challenges with their robust architecture but uses simple correlation for feature matching. This is a crucial module in the network since it encodes the 3D position coordinates to retain geometric information, and mines for the resembling patterns of template in the search area. These target-embedded search area features are the only inputs to RPN.

What could be a better approach to match the template and search area features? While the feature matching module should be permutation invariant to the input point cloud, it should accurately detect the similar features among template and search area, and embed the template features into search area. Self-attention, the core of Transformer block, is capable of detecting even the long-range relationships between the sequences. Our intuition of using Transformers for appearance matching is being the higher-order reasoners, the powerful attention block can efficiently detect the correlated parts among template and search area. Further, through it's input-dependent weights, it could extend to embedding the geometric clues into the search areas. Thus, our main contribution is to depict transformer as an appropriate feature fusion module in 3D object trackers.

2. Related Work

Object Tracking in 2D and 3D. Recently, the Siamese network-based paradigm garnered attention in 2D object tracking. While 2D tracking involves appearance attributes in 2D bounding boxes, 3D trackers determine the position in 3D world in a geometrical perspective. Developing similar networks for point clouds, P2B [11] proposed a pipeline, where PointNet++ backbone extracts seeds from template and search areas. Then, it embeds template seeds into the search area by permutation-invariant feature augmentation. Consequently, a joint 3D target proposal and verification network regresses the potential target centers via Hough Voting to output the bounding box. Later, PTT [12] emphasized that a transformer module plays a non-trivial role in the voting and proposal generation phases. BAT [2] emphasised that the template provided stronger cues about the target and exploited the box-aware and part-aware features of this template in tracking. Recent state-of-the-art architectures like Zheng et al 2022 [1] focused on the motion of object using spatio-temporal features.

Transformers. Further, a few studies leveraged transformers for feature representation and voting in their architecture. One such architecture is GLT-T [8], which proposes a Global-Local Transformer as a voting scheme. Moreover, PTTR [15] and OSP2B [7] supported incorporating self and cross-attention for the fusion of template and target-specific seeds. The state-of-art SwinTrack [6] explored the potential of Swin Transformers as the backbone in 2D object tracking. Essentially, they used the first 3 layers of this transformer and carried out representational learning.

Feature Fusion Modules. Various studies embedded different modules for matching the features. P2B [11] adopted cosine similarity to match appearances and then, employed prediction head of VoteNet [9] to propose regions. Whereas, 3D-SiamRPN [4] adopted a cross-correlation module for feature matching followed by prediction head of VoteNet for detecting regions. These methods perform linear matching, which cannot capture intricate dependencies between the search area and the template.

In our work, we re-implemented some components from P2B [11] and OSP2B [7], to introduce a novel architecture that harnesses Siamese shared backbone for feature extraction and the power of transformers in feature fusion.

3. Transformer-based Siamese 3D SOT

3.1. Overview

Single Object Tracking deals with the problem of localizing a template area in the search area space for each frame in the sequence. This involves extracting the seeds from template and search point clouds using a backbone, followed by the fusion of template and search seeds using a permutation invariant algorithm. Eventually, the network generates target proposals and predicts their target scores, hence outputting both the bounding box and the respective scores. This process involves taking in template point cloud $\mathbb{P}_t \in \mathcal{R}^{\mathbb{N}_t}$ and search area point cloud $\mathbb{P}_s \in \mathcal{R}^{\mathbb{N}_s}$ as input to the shared backbone network and predicts the bounding box $\{x, y, z, \theta\}$ with the highest target-ness score as output.

3.2. Siamese Feature Extractor

The first stage of our architecture has a PointNet++ [10] feature extractor, as the shared backbone. We feed point clouds of template $\mathbb{P}_t$ and search area $\mathbb{P}_s$ to the backbone network and it produces intermediate seeds of template $T = \{t_i\}_{i=1}^{M_1} \in \mathcal{R}^{M_1}$ and search area $S = \{s_i\}_{i=1}^{M_2} \in \mathcal{R}^{M_2}$ respectively as outputs. We chose hierarchical PointNet++ as our backbone since it captures both global and local features to propagate to the next stages. As point clouds doesn't have ordering among the points, this backbone serves as the permutation invariant module by treating each point similarly, irrespective of their position.

$$T = f(P_t)$$
$$S = f(P_s)$$

Where f represents the PointNet++.

3.3. Transformer Feature Fusion

For the feature fusion or matching, we employ an encoder-decoder architecture of transformer network [14]. This network has an encoder with self-attention blocks which consumes template seeds T to output T_{enc}, and similarly a decoder that consumes search seeds S as input to output S_{enc}. Finally, the cross-attention block matches template seeds with that of search area, while merging template geometric clues into the search area.

$$T_{enc} = SelfAttention(T + T_{pos})$$
$$S_{enc} = SelfAttention(S + S_{pos})$$

Prior to passing the seeds as inputs to the transformer, both the template and search seeds are augmented with it's position embeddings $T_{pos} \in \mathcal{R}^{256}$ and $S_{pos} \in \mathcal{R}^{256}$ respectively, that is $[T + T_{pos}] \in \mathcal{R}^{256}$ for template and $[S + S_{pos}] \in \mathcal{R}^{256}$ for search area is passed as inputs. The cross-attention block takes inputs from encoder and applies a feed-forward layer on top of these inputs and considers the resultant as both key and value vectors. Further, it passes the output of decoder through another feed-forward and considers the resulting vector as query.

$$S_{reg} = CrossAttention([T_{enc}], [S_{enc}])$$

3.4. Target Proposal and Verification Network

Taking in the target clue embedded search area seeds $S_{reg} = \{s_j\}_{j=1}^{J}$, each seed s_j is capable of predicting a

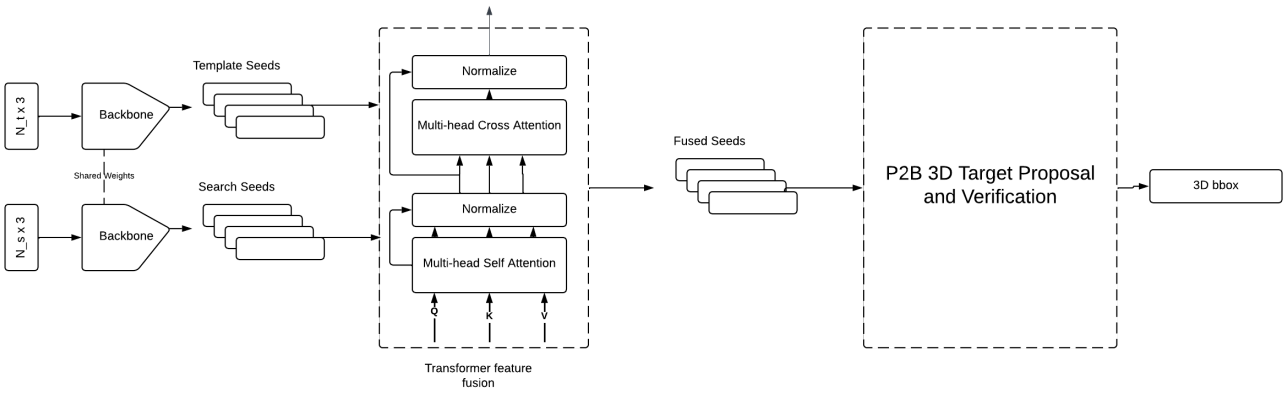

Figure 1. Overview of Transformer-based Tracker Architecture

target center c_j. However, it contains only local clue so multiple neighboring seeds are clustered within a ball of radius R and ensemble of these seeds power the approach to predict a target center while the targetness scores reinforce the confidence.

Following the VoteNet [9], firstly, each seed s_j estimates a target center with the information it contains. That is, it predicts Δx_j, the positional shift of center c_j from the current seed r_j and $\Delta f^t_{r_j}$, the difference in feature vector of the current seed f_{r_j} from the center f_{c_j}. Then, we cluster the seeds within neighborhoods of radius R, $x_k : \|x_j - x_k\|_2 < R$. We subsample a total of K such centroids and regress the targetness scores through an MLP.

$$\{p_j, s^t_j\}^K_{t=1} = MLP(S_{reg})$$

The higher targetness score is, the more likely the template object resides in this region. Finally, the region with the highest target-ness score is predicted as the localization of the template.

3.5. Loss Function

Same as P2B [11], we use a linear combination of 4 components -

1. **Regression loss** ($\mathcal{L}_{reg}$): L1 loss of each predicted target's positional shift verses it's ground truth's positional shift for each seed.

$$\mathcal{L}_{reg} = \Sigma^J_{j=1} \|\Delta x_j - \Delta gt_j\|$$

2. **Proposal loss** ($\mathcal{L}_{pro}$): Binary cross-entropy loss of the predicted targetness scores $\hat{s}_j$, while the ground truth scores are s_j.

$$\mathcal{L}_{pro} = \Sigma^J_{j=1} - s_j \log(\hat{s}_j)$$

3. **Classification loss** ($\mathcal{L}_{cla}$): Binary cross-entropy loss of the predicted class.

4. **Box loss** ($\mathcal{L}_{box}$): Huber loss for the predicted box parameters.

Overall, loss is

$$\mathcal{L} = \mathcal{L}_{reg} + \mathcal{L}_{pro} + \mathcal{L}_{cla} + \mathcal{L}_{box}$$

4. Experiments

4.1. Experimental Setup

4.1.1 Dataset

We used the LiDAR point cloud data from the KITTI Velodyne dataset [5] for our case. There are a total of 21 sequences, of which we are using sequences 0-16 for training, 17-18 for validation, and 19-20 for testing. We trained our model on the Car, Pedestrian, Van, and Cyclist categories.

4.1.2 Evaluation Metrics

We employ success and precision metrics using One Pass Evaluation. Success is the intersection over the Union (IoU) between the predicted and ground-truth bounding box, that is, the overlap of the predicted box with the ground-truth box. Precision is the distance of the center of the predicted bounding box from the ground-truth box, with a threshold of up to 2 meters.

4.1.3 Implementation Details

From the template and search area, we randomly sampled 512 and 1024 point from the point cloud as mentioned in P2B [11]. The ways in which samples are generated differ in training and testing. During training as P2B, we considered the search area by applying random offsets on previous gt boxes and extending them by 2 meters in all directions. For template area, we applied random offset on previous gt

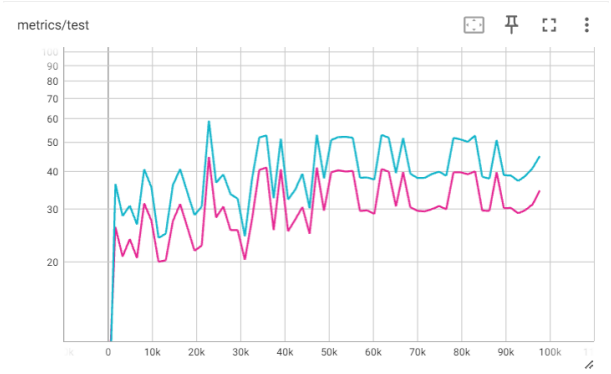

Figure 2. Precision and Recall of validation set evaluated after each epoch during training. The blue line depicts precision metric and pink line for success.

Model	Precision				
	Car	Van	Pesdestrian	Cyclist	Mean
	6,424	1,248	6,088	308	14,068
1. P2B [11]	72.8	48.4	49.6	44.7	60.0
2. OSP2B [7]	82.3	66.2	85.1	90.5	82.3
3. LTTR [3]	77.1	48.4	56.8	89.9	65.8
4. Ours	72.0	52.8	63.9	46.5	66.0

Table 1. Comparison of our precision results on different categories with SOTA models

template area. During testing, we use only the previous predicted and first box of point clouds and merge them to perform the sampling for template. Whereas, for search area, we use previous result box and extended it by 2 meters.

The PointNet++ [10] backbone implemented for feature extraction has 3 abstraction layers with 64 template seeds, 128 search seeds and 256 feature dimensions are generated. We followed P2B settings for our backbone. The Attention blocks in the transformer have 1 attention head, with dimensions of 128 for the query and key vectors and 256 for the value vector. The Feedforward network following the attention block has 2 linear layers of dimensions 128 and 256 respectively, followed by an instance normalization block. Both the Encoder and Decoder of the transformer have 1 layer each. The architecture overall has 2.1m parameters with the transformer network alone has 1.2m parameters.

4.1.4 Experimental Setup

The Codebase was set up on GCP with a Nvidia L4 GPU instance of 24GB RAM, 8vcpus of 32GB RAM as a whole, and a dedicated persistent storage of 100 GB. The model was trained for 60 epochs and we saw convergence around epochs 20-30 for the Car category and 10-20 for the rest.

Model	Success				
	Car	Van	Pesdestrian	Cyclist	Mean
	6,424	1,248	6,088	308	14,068
1. P2B [11]	56.2	40.8	28.7	32.1	42.4
2. OSP2B [7]	67.5	56.3	53.6	65.6	60.5
3. LTTR [3]	65.0	35.8	33.2	66.2	48.7
4. Ours	56.8	45.6	35.7	34.1	45.5

Table 2. Comparison of our success results on different categories with SOTA models

4.2. Comparison with Baselines

We compare our results with P2B [11], OSP2B [7] and LTTR [3] baselines as our architecture is inspired by P2B and OSP2B and incorporates transformers. We present our results in Table 1.

We compare our results on categories like Car, Pedestrian, Van and Cyclist. It can be seen that our model outperforms the P2B model by 6% on an average precision and 3.1% on average success. This can be solely attributed to the Transformer Feature Fusion module. Our model outperforms P2B in the Pedestrian category by a large margin, showcasing the robustness on Non-Rigid categories. It could be observed in Table 1 that our model degraded a little on the Car category. Nevertheless, with extensive hyperparameter tuning, we are sure to surpass P2B.

From tables 1 and 2, it can also be seen that our model performs equally good when compared with LTTR [3], but has degraded a lot when comparing with OSP2B. This is because, OSP2B uses a parameter-efficient parallel predictor RPN network, that performs better than our baseline P2B RPN. OSP2B rather than traditional Euclidean space thresholded labeling, follows shape-adaptive labels, which consider spatial distribution of proposals inside objects for centerness score. OSP2B also uses foreground and background information during the target classifier stage, thus efficiently scoring foreground proposals and penalizing the background ones. Furthermore it can be seen that our model degrades by a large margin on the cyclist category compared to LTTR, this could be attributed to the proficiency of LTTR on smaller size categories and less training data.

4.3. Ablation Studies

Similar to P2B, we have analyzed our model by trying out different approaches for the search and template area point cloud sampling. Table 3 reports the precision and success results using different search area generation methods. Here, the Previous result uses the model's bounding box outputs from $(i-1)^{th}$ frame in generating search area for i^{th} frame. Precisely, we take the model's $(i-1)^{th}$ output and extend it by 2 meters in all directions and consider it as search area for i^{th} input. Similarly, Previous GT uses $(i-1)^{th}$ GT bounding boxes for i^th search area and Cur-

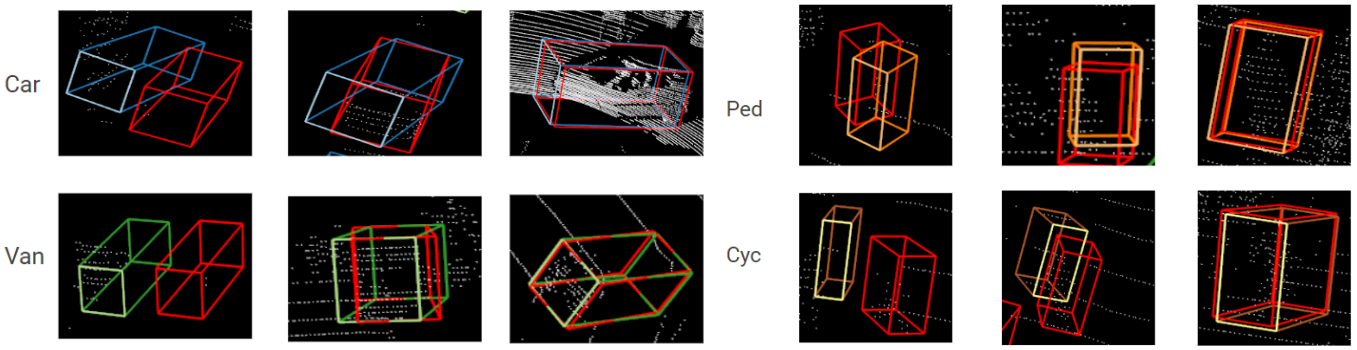

Figure 3. For each of the categories, the first column is an instance of sparse point clouds, the second column is an instance of occluded ones and the third column is a perfect prediction

Metric	Method	Previous result	Previous GT	Current GT
Success	P2B	56.2	82.4	84.0
	Ours	56.8	82.0	84.7
Precision	P2B	72.8	90.1	90.3
	Ours	72.0	89.8	90.7

Table 3. Comparison of our success results for different search areas taken into consideration during prediction.

Metric	Method	First GT	Prev Result	First and Prev	All Prev results
Success	P2B	46.7	53.1	56.2	51.4
	Ours	52.5	55.2	56.8	52.7
Precision	P2B	59.7	68.9	72.8	66.8
	Ours	66.2	70.0	72.0	66.9

Table 4. Comparison of our success results for different template areas taken into consideration during prediction.

rent GT uses i^{th} bounding box in generating its search area respectively. It can be observed that our model performs similarly to P2B in all the cases.

For template area generation, Table 4 reports the results on different methods. Here, First GT uses the ground truth bounding box of the first frame for template point cloud sampling. Previous Result uses $(i-1)^{th}$ output bounding box in generating i^{th} template. area. The first and Previous result uses both the GT bounding box of the first frame and $(i-1)^{th}$ output bounding box. All Previous results uses all the previous outputs of the sequence for template area sampling. It can be seen from Tables 3 and 4 that in most of these approaches, our model outperforms P2B due to robust appearance matching. Specifically, it outperforms P2B by a large margin when using current ground truth of search area or first ground truth of template.

4.4. Qualitative analysis

We have analyzed the cases where our model performs better and worse. It was observed that for the point clouds having a decent number of points, or unoccluded point clouds yield good predictions. This is evident from the third and sixth columns of Figure 3.

In the case of Sparse point clouds and point clouds with occlusions, the model is not able to predict the bounding boxes accurately. This can be seen in the first, second, fourth and fifth columns of Figure 3. This issue can be handled by using a point cloud completion algorithm, in order to generate more points for sparse point clouds and occluded regions.

We have also analyzed the speed of our model, in order to judge its use case in real-life scenarios like Self-driving cars. Our model runs at a mean speed of 20.5 FPS on test run for car category on NVIDIA L4 GPU.

5. Conclusion

Object Tracking has shown tremendous potential in the last few years. Particularly with the introduction of transformers into the 3D Vision models. Our methodology unleashes the potential of transformers as feature fusion networks in point cloud-based 3D object tracking. Our method outperforms the state-of-the-art model Point-to-Box by 6% average precision and 3.1% average success.

Future Considerations. We believe this idea of fusing features using attention blocks can be extended to Multiple Object Tracking to get fruitful results. Experiments also show that our model fails to deal with sparse and occluded point clouds. This shortcoming can be tackled by using a point cloud completion algorithm. Furthermore, pairwise 2D-3D annotations can be used in learning the missing information in point cloud from stereo or monocular image data. This could not only make the model robust but also improve results.

References

[1] Haiming Zhang Baoyuan Wang Shenghui Cheng Shuguang Cui Zhen Li Chaoda Zheng, Xu Yan. Beyond 3d siamese

Figure 4. A picture of the tracked point cloud(left) and its corresponding camera image(right).

tracking: A motion-centric paradigm for 3d single object tracking in point clouds. 2022. 2

[2] Jiantao Gao Weibing Zhao Wei Zhang Zhen Li Shuguang Cui Chaoda Zheng, Xu Yan. Box-aware feature enhancement for single object tracking on point clouds. 2021. 2

[3] Yubo Cui, Zheng Fang, Jiayao Shan, Zuoxu Gu, and Sifan Zhou. 3d object tracking with transformer, 2021. 4

[4] 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*, 21(4):4995–5011, Feb. 2021. 2

[5] 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*, pages 3354–3361, 2012. 3

[6] Zhipeng Zhang Yong Xu Haibin Ling Liting Lin, Heng Fan. Swintrack: A simple and strong baseline for transformer tracking. 2021. 2

[7] Jiahao Nie, Zhiwei He, Yuxiang Yang, Zhengyi Bao, Mingyu Gao, and Jing Zhang. Osp2b: One-stage point-to-box network for 3d siamese tracking, 2023. 2, 4

[8] Jiahao Nie, Zhiwei He, Yuxiang Yang, Mingyu Gao, and Jing Zhang. Glt-t: Global-local transformer voting for 3d single object tracking in point clouds, 11 2022. 2

[9] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J. Guibas. Deep hough voting for 3d object detection in point clouds, 2019. 2, 3

[10] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space, 2017. 2, 4

[11] Haozhe Qi, Chen Feng, Zhiguo Cao, Feng Zhao, and Yang Xiao. P2b: Point-to-box network for 3d object tracking in point clouds, 2020. 2, 3, 4

[12] Jiayao Shan, Sifan Zhou, Zheng Fang, and Yubo Cui. Ptt: Point-track-transformer module for 3d single object tracking in point clouds. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1310–1316, 2021. 2

[13] Jesus Zarzar Silvio Giancola and Bernard Ghanem. Leveraging shape completion for 3d siamese tracking., 2019. 1

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 2

[15] Changqing Zhou, Zhipeng Luo, Yueru Luo, Tianrui Liu, and Liang Pan. Pttr: Relational 3d point cloud object tracking with transformer, 2022. 2