ODTFormer: Efficient Obstacle Detection and Tracking with Stereo Cameras Based on Transformer

Tianye Ding[1*], Hongyu Li[2*], and Huaizu Jiang[1]

I. Introduction

Obstacle detection and tracking represents a safety-critical challenge across various domains, including robot autonomous navigation [1–5] and self-driving vehicles [6–9]. For instance, a service robot needs to detect people and pillars surrounding it, track their motions (if any), or even predict their future trajectories to avoid collision. Accurate obstacle detection and tracking are crucial components of autonomous navigation systems, particularly in state-based frameworks, to ensure collision-free navigation [10–13]. Recent research efforts focus on using low-cost visual sensors for obstacle perception to improve affordability [10, 14–17] compared with costly ones (*e.g.*, LiDAR). In this paper, we concentrate on a specific line of research employing stereo cameras, which offer higher 3D perception accuracy, extended sensing range, and enhanced agility for robots compared to monocular-based systems [18, 19].

In this paper, we present ODTFormer, a Transformer-based [20] model, to address the aforementioned obstacle detection and tracking challenges. Unlike existing stereo-based models, which construct a pixel-wise cost volume for disparity matching [14, 21–24], we propose to use deformable cross-attention [25] from 3D voxel queries to 2D stereo image features to compute the matching cost. Compared with the pixel-wise cost volume used in [14], we directly construct it in the 3D space, conforming better to the scene geometry. More importantly, our approach disentangles dataset specifics from the model design and thus shows better generalization than [14]. The cost volume is then processed by a U-Net decoder to produce occupancy voxel grids progressively for efficiency purposes.

To account for environmental dynamics, we introduce a novel obstacle tracker to capture the motions of the scene by matching similar voxels between two consecutive frames. We integrate physical constraints into the voxel tracking by setting a volumetric bound for each voxel when searching for its corresponding voxel in the next frame, which improves both accuracy and efficiency.

To validate the effectiveness of our approach, we conduct comprehensive evaluations on DrivingStereo [26] and KITTI [27] datasets. In the obstacle detection task, we demonstrate a significant performance improvement compared to prior works [14, 21–24]. Additionally, our approach exhibits greater generalization across various camera parameters and resolutions than [14]. In the obstacle tracking task, we compare scene flow-based approaches and show that our approach can achieve comparable results to state-of-the-art methods while needing much lighter computation cost. Detailed ablation studies are conducted to validate different design choices. In summary, our contributions are:

- A novel 3D cost volume construction method based on the deformable cross-attention [25], which better conforms to the scene geometry and generalizes well to different camera parameters and image resolutions.
- A novel obstacle tracking method by matching the voxels across two consecutive frames, which can be jointly optimized with the detection module in an end-to-end manner.
- Experimental results show that our approach achieves better or comparable detection and tracking accuracy to state-of-the-art methods while being efficient and running fast at the same time (20fps on the KITTI resolution of 370×1224 on an RTX A5000 GPU without careful postprocessing, *e.g.*, quantization, pruning).

II. Proposed Method

In this paper, we present a novel approach to tackle both obstacle detection and tracking problems. For the obstacle detection task, our model estimates a voxel occupancy grid in front of the stereo cameras, defining this area as our region of interest (ROI). Subsequently, our model efficiently tracks the motion of each detected obstacle by matching voxels across two consecutive frames.

A. Overview

Obstacle detection. Following StereoVoxelNet [14], we formulate obstacle detection as occupancy detection based on stereo images. As shown in Fig. 1 (a), our detection model consists of three modules: i) 2D image feature extraction, ii) cost volume construction via cross-attention, and iii) occupancy decoding.

1) 2D image feature extraction: Given the rectified stereo images $\mathbf{I}^L$ and $\mathbf{I}^R$, we employ a lightweight EfficientNet-B0 [28] backbone enhanced by the FPN [29] to extract multi-scale image features, denoted as $\mathbf{F}_i^L$ and $\mathbf{F}_i^R$, respectively. Here the subscript $i \in \{1, 2, 3, 4\}$ indicates the level (scale) of the feature maps unless explicitly mentioned otherwise. With the input image resolution denoted as $H \times W$, the multi-scale feature maps reduce the resolutions to $\{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}\}$, with the feature channel dimension of D for all the levels.

This research is supported by the National Science Foundation under Award Number IIS-2310254.

*Equal contribution.

[1]Northeastern University, Boston, MA, 02115. {ding.tian, h.jiang}@northeastern.edu

[2]Brown University, Providence, RI, 02912. hli230@cs.brown.edu

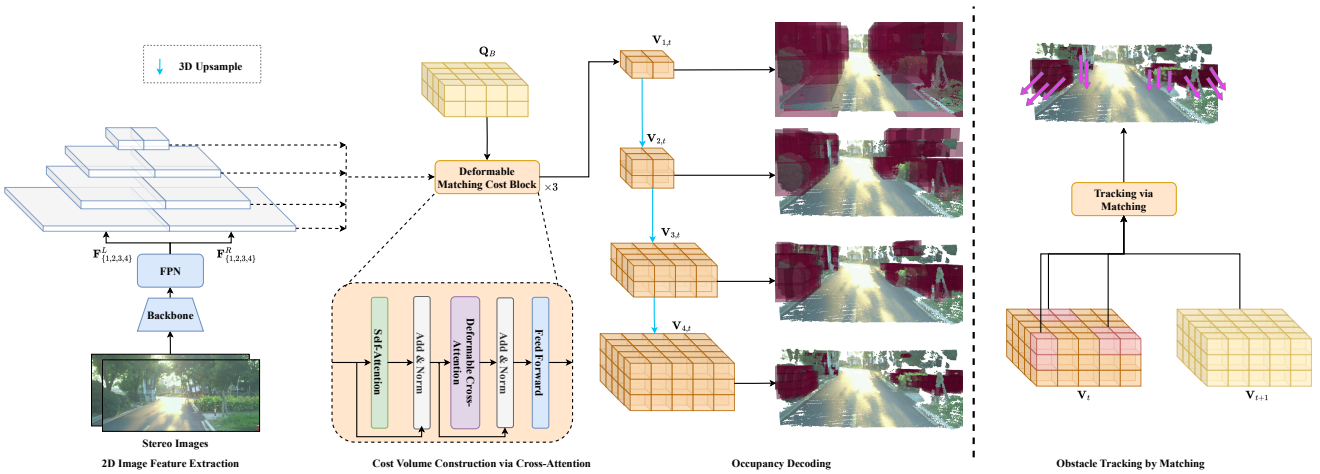


Fig. 1: **Illustration of the overall architecture design. Left:** For obstacle detection, we first extract multi-scale 2D feature maps [28, 29] for each of the stereo images. We then encode the voxels in the ROI to cross-attend to the image features to compute the matching cost through our novel cost volume construction method. Such a cost volume is directly constructed in the 3D space, which conforms better to the scene geometry, disentangles dataset specifics from model design, and thus generalizes well. It is then progressively decoded into occupancy voxel grids. **Right:** For obstacle tracking, we cast it as a matching problem by finding the correspondences of voxels across two consecutive frames, where we incorporate physical constraints to improve both the accuracy and efficiency. Both the detection and tracking modules can be jointly optimized in an end-to-end manner and run efficiently.

2) Cost volume construction via cross-attention: A crucial difference of our proposed approach from [14] is our novel cost volume construction module design, where we employ cross-attention from 3D voxel queries to 2D stereo image features. Compared with the pixel-wise cost volume used in [14], we directly construct it in the 3D space, conforming better to the scene geometry. More importantly, our approach disentangles dataset specifics from the model design and thus shows better generalization than [14]. We will introduce this module in detail in Sec. II-B.

3) Occupancy decoding: Similar to [14], as the final step in the obstacle detection module, we estimate the voxel occupancy in a coarse-to-fine-grained manner. We utilize a 3D U-Net decoder to upsample the cost volume gradually. Specifically, in the time step t, our model outputs four-level coarse-to-fine voxel occupancy volumes $\mathbf{V}_{i,t}$ with the constant ROI size of $s_x \times s_y \times s_z$ meters,

$$\mathbf{V}_{i,t} = \{0,1\}^{2^{i-1}(n_x \times n_y \times n_z)}, i \in \{1,2,3,4\}, \quad (1)$$

where i denotes the level number. n_x, n_y, and n_z denote the number of voxels in the x, y and z axes, respectively. Each voxel is classified as either empty (denoted by 0) or occupied (denoted by 1) and has a side length of l_i which is decreased by half when the level number increases to keep grid metric size $s_{\{x,y,z\}}$ constant, $l_{i+1} = \frac{1}{2} l_i$.

Obstacle tracking by matching. After getting obstacle detection results in the form of occupied voxels, we'd like to track each of them for a better understanding of scene dynamics. To this end, we find correspondences of occupied voxels across two consecutive time steps via *global matching*, as shown in Fig. 1 (the right part). Such tracking information provides critical cues of dynamic objects for visual navigation. We introduce this part in detail in Sec. II-C.

Finally, we describe how the entire model is trained in Sec. II-D.

B. Cost Volume Construction via Cross-Attention

Cost volume plays a critical role for stereo-based obstacle detection methods [14, 21–24]. Conventional cost volume construction involves computing *pixel-wise* matching costs across a predefined range of disparity/depth levels [14]. However, this approach encounters significant drawbacks. First of all, the dimensions of the cost volume do not directly correlate with the resolution of the voxel grid. To solve this issue, the cost volume is compressed into a single latent vector and subsequently resized into the desired ROI voxel volume in [14]. This process inevitably leads to the loss of geometry information, which is critical for accurate obstacle detection. Furthermore, the hallucination of the latent vector to a 3D voxel volume assumes a fixed ROI dimension, prohibiting its generalization to a different setting[1]. In addition, the camera intrinsics and extrinsics are directly embedded into the construction of the cost volume. Such entanglement of the dataset specifics and model design restricts its applicability across different domains and setups, making generalization challenging, especially in scenarios with varying camera parameters and image resolutions.

Instead of constructing a cost volume at the pixel level, we propose a novel approach to build it in the 3D space directly

[1]It is in analogy to the existence of the fully connected layers in AlexNet. As a result, the model only accepts an input image of a fixed dimension.

via cross-attention from voxel queries to image features. In a nutshell, we first encode each voxel in the ROI, which will then attend to image features via deformable attention [25]. Compared with the conventional cost volume [14], our approach better conforms to the scene geometry and disentangles dataset specifics and model design, allowing better generalization. We explain different components in detail as follows. For brevity, we omit the time step index t in this section.

Encoding of voxels in the ROI. We partition the ROI into a set of voxels $\mathbf{Q}_B$ with a dimension of $n_x \times n_y \times n_z$ to balance the computation burden and spatial resolution of the cost volume. Each voxel in $\mathbf{Q}_B$ is encoded as the Fourier positional encoding [20] of its normalized centroid coordinates (between 0 and 1), which is fed into a multi-layer perceptron (MLP) to get the encoding $\mathbf{q}$. It is further enhanced by the pixel-aligned features $\mathbf{f}$, $\mathbf{q} = \mathbf{q}+\mathbf{f}$ as in [30]. $\mathbf{f}$ is defined as

$$\mathbf{f} = \frac{1}{2}(\mathbf{F}_4^L(\mathcal{P}(p, \theta^L)) + \mathbf{F}_4^R(\mathcal{P}(p, \theta^R))), \tag{2}$$

where p is the voxel's centroid. $\mathcal{P}(p, \theta^L)$ and $\mathcal{P}(p, \theta^R)$ denote the projections of the voxel centroid onto the left and right images, respectively, according to their projection matrices θ^L and θ^R. Bilinear interpolation is used to sample features from the image feature maps $\mathbf{F}_4^L$ and $\mathbf{F}_4^R$.

Deformable cross-attention for matching cost. We leverage the deformable matching cost (DMC) block to update voxel queries with calculated matching costs for each query. Our DMC block is built on top of the transformer decoder block [20] with deformable attention [25].

Specifically, for the voxel centered at p, we first sample points on multi-scale feature maps $(\mathbf{F}_i^L, \mathbf{F}_i^R)$ (similar to the pixel-aligned features $\mathbf{f}$ but on multi-scale features instead of just the last level) around the projected points of the voxel centroid p with the offset δ, respectively. δ is generated by a learnable 3D offset sampler g [25] based on the voxel's encoding $\mathbf{q}$, $\delta = g(\mathbf{q})$, where g is an MLP plus a sigmoid output layer (scaled by the 3D voxel size of the coarsest output, see experiment settings in Sec. III-A). We then directly concatenate the sampled features along the channel dimension and employ a scale-specific MLP to compute the cross-view matching cost $\mathbf{c}_i$:

$$\begin{aligned}\mathbf{c}_i(p, \mathbf{q}, \mathbf{F}_i) = \texttt{MLP}_i(\mathbf{F}_i^L(\mathcal{P}(p+\delta, \theta^L)) \oplus \\ \mathbf{F}_i^R(\mathcal{P}(p+\delta, \theta^R))),\end{aligned} \tag{3}$$

where $\oplus$ denotes the feature concatenation. $\texttt{MLP}_i$ has two linear layers with batch normalization and ReLU activation in between. Since we project the same 3D sample point onto multi-scale feature maps, the resultant multi-scale matching costs for this 3D point can be aggregated by concatenating along the channel dimension and employing an MLP to compute the voxel matching cost $\mathbf{c}$:

$$\begin{aligned}\mathbf{c}(p, \mathbf{q}, \mathbf{F}_{\{1,2,3,4\}}) = \texttt{MLP}(\mathbf{c}_1(p, \mathbf{q}, \mathbf{F}_1) \oplus \mathbf{c}_2(p, \mathbf{q}, \mathbf{F}_2) \oplus \\ \mathbf{c}_3(p, \mathbf{q}, \mathbf{F}_3) \oplus \mathbf{c}_4(p, \mathbf{q}, \mathbf{F}_4)),\end{aligned} \tag{4}$$

where $\texttt{MLP}$ has the same layer configuration as $\texttt{MLP}_i$. Instead of using a single offset, multiple ones are usually generated to enhance the computing of the matching cost, and different matching costs for different samples are aggregated as follows.

$$\mathbf{C}_B(p, \mathbf{q}, \mathbf{F}) = \sum_{s=1}^{N_s} \mathbf{A}_s \mathbf{W}_s \mathbf{c}(p, \delta_s, \mathbf{F}_{\{1,2,3,4\}}), \tag{5}$$

where N_s is the total number of sampling points. δ_s denotes the s-th learned offset. $\mathbf{A}_s \in [0, 1]$ is the learnable weights for the cost generation and $\mathbf{W}_s \in \mathbb{R}^{D \times D}$ are learnable parameters. We can get a cost volume $\mathbf{C}_B \in \mathbb{R}^{n_x \times n_y \times n_z \times D}$ by computing the matching cost for all voxels in $\mathbf{Q}_B$.

Deformable cross-attention has been used in previous works [6–9] to aggregate multi-view and multi-scale features. But instead of simply averaging them as in existing work, we leverage the geometric constraints between two stereo images and use the deformable attention to construct a cost volume in 3D, leading to better accuracy, as evidenced by our ablation studies.

The cost volume $\mathbf{C}$ will be processed to output occupancy voxel grid $\mathbf{V}_{i,t}$ progressively using a U-Net decoder [14].

C. Voxel Tracking by Matching

Given the estimated occupancy grid $\mathbf{V}_t$ and $\mathbf{V}_{t+1}$ at the time step t and $t+1$, respectively[2], our goal is to find the 3D motion vector of each voxel in $\mathbf{V}_t$ by finding its correspondence in $\mathbf{V}_{t+1}$. Apparently, we do not need to worry about unoccupied voxels. For the occupied ones, we compute their cosine similarities $\mathbf{S}_t \in [-1, 1]^{N_t \times N_{t+1}}$ by comparing their feature representations in the U-Net decoder right before the output layer. N_t and N_{t+1} denote the number of occupied voxels in $\mathbf{V}_t$ and $\mathbf{V}_{t+1}$, respectively. Optionally, we can match all voxels in $\mathbf{V}_{t+1}$ regardless of their occupancy status. We will ablate this design choice in the experiment section. The matching distribution of the occupied voxels can then be computed as

$$\mathbf{P}_t = \text{softmax}(\tau \mathbf{S}_t), \tag{6}$$

where τ is a learnable logit scale. The softmax is applied in a row-wise manner. Motion vectors $\mathbf{M}_t$ of the occupied voxels can be computed as

$$\mathbf{M}_t = \mathbf{P}_t \mathbf{G}_{t+1} - \mathbf{G}_t, \tag{7}$$

where $\mathbf{G}_t \in \mathbb{Z}^{N_t \times 3}$ denotes the 3D coordinates of each voxel's centroid in $\mathbf{V}_t$.

In practice, the motions of the occupied voxels are bounded by physical constraints. Instead of naively comparing all possible pairs of occupied voxels, we set a volumetric bound surrounding each occupied voxel in $\mathbf{V}_t$ and only measure the similarities for voxels contained in this boundary when computing $\mathbf{S}_t$. For the voxels outside of the boundary, the similarity score is set to be $-\infty$. Specifically, assuming

[2]They are the output of the last layer in the U-Net decoder, $\mathbf{V}_{4,t}$ and $\mathbf{V}_{4,t+1}$. For brevity, we simply use $\mathbf{V}_t$ and $\mathbf{V}_{t+1}$ in this section when the context is clear.

that a stereo image sequence is captured at least f frames per second (FPS) and the obstacles have a smaller relative velocity of than v meters per second to us, the maximum displacement of any trackable voxel across two frames is $d = \frac{v}{f}$. With the voxel at the center of the volumetric region, the volume bound's dimensionality can be calculated as $(2\lceil \frac{d}{l_4} \rceil + 1) \times 3 \times (2\lceil \frac{d}{l_4} \rceil + 1)$ for x, y and z axes, where l_4 is the voxel side length in the last level of the U-Net decoder. The maximum displacements along the y-axis (pointing downward the ground plane) in both directions have been set to 1 to ignore drastic vertical movements within the driving scenario. As we will see in the experiment section, it not only eases the computation burden but also leads to better obstacle tracking accuracy.

D. Model Training

We optimize the obstacle detection model using the weighted sum of the Intersection of Union (IoU) loss $\mathcal{L}_D$ over four U-Net decoder levels:

$$\mathcal{L}_D = \sum_{j=1}^{4} w_j (1 - \texttt{IoU}(\mathbf{V}_{j,t}, \bar{\mathbf{V}}_{j,t})), \tag{8}$$

where $\bar{\mathbf{V}}_{j,t}$ denotes the ground-truth voxel grid at level j. w_j is the weight assigned to the U-Net decoder level, and we set w_1 to w_4 as $[0.30, 0.27, 0.23, 0.20]$ empirically.

The tracker is trained using the endpoint error (EPE) loss, which is the average of element-wise L2 distances between all estimated and ground-truth 3D motion vectors. We note, however, that optimizing the loss for detected occupied voxels leads to degenerated performance because the detection results may be wrong. Instead, we define the loss for *all* voxels regardless of their detection results. The loss is defined as

$$\mathcal{L}_T = \frac{1}{N'_t} \sum_{k=1}^{N'_t} \left|\left| \mathbf{M}'_t[k] - \mathbf{M}_t[k] \right|\right|_2, \tag{9}$$

where $\mathbf{M}'$ contains the motion vectors for all the voxels. For the k-the voxel, if it is detected as occupied, $\mathbf{M}'[k] = \mathbf{M}[k]$. Otherwise, $\mathbf{M}'[k] = \mathbf{0}$. $\bar{\mathbf{M}}'$ denotes the ground-truth motion vectors.

The ODTFormer is trained progressively. We first train the detection module to obtain reliable detection results. We then jointly train the detection and tracking modules together, where the feature representations of the voxels are refined to encode the spatio-temporal cues effectively.

III. Experiments

In this section, we evaluate the performance of ODTFormer against various baselines on both obstacle detection (Sec. III-A) and obstacle tracking (Sec. III-B) tasks.

A. Obstacle Detection

We optimize our voxel occupancy prediction module using the DrivingStereo dataset [26], which contains over $180,000$ stereo pairs for the driving scenario with a resolution of 400×880. During preprocessing, we augment the training images using color jittering.

TABLE I: **Quantitative obstacle detection results on the DrivingStereo testing set.** The best result is bold and the second-best result is underlined. We use an RTX A5000 GPU for measuring the inference speed.

Method	Level	CD (meters) ↓		IoU (%) ↑		MACs ↓	Params ↓	FPS ↑
		15.0m	30.0m	15.0m	30.0m			
2D-MSNet [21]	4	9.72	21.01	18.14	9.38	91.74G	<u>2.23M</u>	6.7
3D-MSNet [21]	4	5.96	13.14	21.11	11.96	414.59G	**1.77M**	3.6
ACVNet [24]	4	8.43	18.82	18.32	13.59	801.33G	7.11M	4.4
Lac-GwcNet [23]	4	7.19	<u>12.90</u>	29.84	17.25	777.19G	9.37M	4.8
CFNet [22]	4	9.79	18.65	21.21	10.35	456.14G	22.24M	4.9
StereoVoxelNet [14]	1	4.54	5.12	80.07	71.94	<u>16.03G</u>	4.58M	**21.8**
	2	3.07	4.34	70.08	56.39			
	3	2.75	6.87	51.80	37.14			
	4	3.15	17.20	38.27	21.92			
StereoVoxelNet w/ EfficientNet	1	4.36	5.17	80.26	71.93	**14.83G**	10.46M	20.2
	2	2.81	4.28	70.55	56.32			
	3	2.49	5.95	52.59	37.83			
	4	<u>2.90</u>	13.56	<u>38.69</u>	<u>23.04</u>			
ODTFormer (**Ours**)	1	**3.09**	**4.13**	**85.25**	**76.75**	25.05G	6.14M	<u>21.7</u>
	2	**1.92**	**3.22**	**76.10**	**62.69**			
	3	**1.64**	**3.82**	**57.94**	**43.79**			
	4	**1.87**	**8.72**	**41.62**	**26.62**			

We set our ROI (defined in Sec. II-A.3) with respect to the left camera as $30m$ ahead, $8m$ to the left, $10m$ to the right, and $3m$ to the top and bottom. We set the voxel sizes to be $3m$, $1.5m$, $0.75m$, and $0.375m$ for each level, and the finest level will have a voxel grid of size $n_x \times n_y \times n_z = 48 \times 16 \times 80$. We precompute and store the precise voxel occupancy grid ground truth by cropping our ROI and removing all voxels that are $1.5m$ under the viewpoint on the y-axis (ground plane).

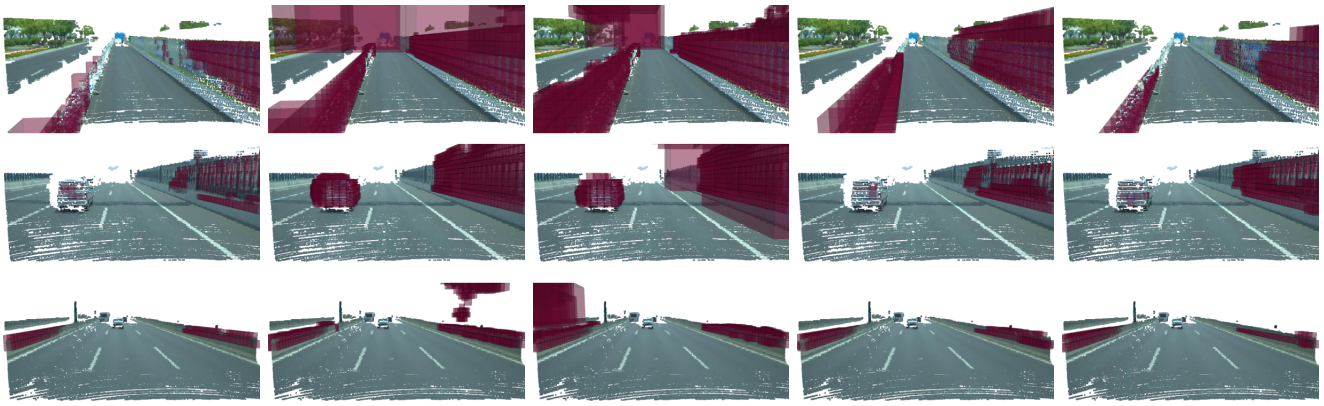
The network is trained for 20 epochs with a batch size of 16 using AdamW [31] optimizer. We set the initial learning rate as 0.0001 and decrease it each epoch using the Cosine Annealing learning rate scheduler with a minimum learning rate of 1×10^{-8}.

Evaluation metrics. We choose IoU and Chamfer Distance (CD) as our evaluation metrics. IoU measures the intersection between the ground truth and the predicted voxel occupancy grids. CD measures the distance between the point clouds (by presenting the centroids of occupied voxels as point clouds). We report the output accuracies with the $15.0m$ range (half of the ROI depth) and the $30.0m$ range (entire ROI depth) separately, which represents easier and harder-to-identify obstacles.

Besides the accuracy metrics, we also consider computational cost matrics: multiply-accumulate operations (MACs), the number of model parameters (Params), and frame-per-second (FPS).

Baseline methods. We compare obstacle detection performance against two types of seminal works: stereo-based depth estimation and voxel occupancy prediction. For depth estimation models, following the prior works [10, 14], we first convert the estimated depth map into a point cloud and subsequently voxelize it into a voxel occupancy grid. We include MobileStereoNet (MSNet, both 2D and 3D version) [21], ACVNet [24], Lac-GwcNet [23] and CFNet [22]. For the occupancy prediction approach, we compare against StereoVoxelNet [14]. To rule out the effect of feature extraction, we also ablate the original feature extraction backbone of StereoVoxelNet with EfficientNet (same as ours).

As shown in Tab. I, ODTFormer significantly outperforms



Ground Truth | 3D-MSNet [21] | Lac-GwcNet [23] | StereoVoxelNet [14] | **Ours**

Fig. 2: Visual results of obstacle detection on the DrivingStereo dataset.

TABLE II: **Quantitative obstacle tracking results on the KITTI 2015 scene flow dataset.** RAFT-3D MACs are evaluated including MobileStereoNet. The best result is bold, and the second-best result is underlined.

Method	EPE (meters) ↓	Foreground EPE (meters) ↓	MACs ↓
PWOC-3D [32]	0.043	1.23	N/A
SENSE [33]	0.048	1.18	<u>284.73G</u>
RAFT-3D (2D-MSNet) [34]	<u>0.025</u>	<u>0.91</u>	649.46G
RAFT-3D (3D-MSNet) [34]	0.028	**0.87**	1295.16G
ODTFormer (**Ours**)	**0.021**	1.08	**64.47G**

all other approaches, particularly depth-based methods [21–24], in terms of both IoU and CD. Although ODTFormer exhibits higher MACs compared to StereoVoxelNet, we achieve similar FPS. This discrepancy arises because we do not construct cost volumes iteratively based on stereo matching, enabling better parallelism.

B. Obstacle Tracking

We finetune ODTFormer for voxel tracking using the synthetic SceneFlow Driving dataset [35] and the real-world KITTI 2015 dataset [27], which contains over $4,000$ stereo image sequences with the resolution of 540×960 and 200 stereo images with the resolution of 370×1224, respectively. We resize the images and camera parameters to have the same resolution 400×880 as in DrivingStereo, then apply the same data augmentation as obstacle detection and voxelize the scene flow by averaging the scene flow vectors of the point cloud within occupied voxels. We set the FPS and velocity thresholds for the matching boundary as $f = 26$ FPS and $v = 33.3m/s$ (or 120 km/h). The volumetric region of each voxel during tracking is set to be the surrounding $9 \times 3 \times 9$ voxels along x, y, z axes.

We randomly split the KITTI training dataset into a custom training set and a validation set using an 8:2 ratio. We first train the network for 20 epochs with a batch size of 8 on the SceneFlow Driving dataset, then finetune on the KITTI dataset for 50 epochs.

Evaluation metrics. We employ EPE and foreground EPE to assess our model's performance in voxel flow estimation. EPE quantifies the Euclidean distance between the ground-truth flow vector and the estimated flow vector. Foreground EPE is the EPE loss measured upon those voxels classified as occupied within the ground truth so that the measurement considers both occupancy detection and flow estimation accuracies.

Baseline methods. We compare against stereo-based scene flow estimation methods: SENSE [33], PWOC-3D [32], and RAFT-3D [34] with MobileStereoNet (MSNet) [21] for depth estimation. We use their model weights trained on the *entire* KITTI 2015 training set, including our custom validation set, to evaluate all the methods. We first estimate the scene flow given stereo image pairs from consecutive frames, then voxelize it using the same process as we generate the ground truth (Sec. III-B).

The quantitative results are evaluated using our custom validation set within the KITTI 2015 training data since our model cannot be directly evaluated through the KITTI test submission. The results are summarized in Table II. Notably, we achieve comparable performance to RAFT-3D using only ten-fold or twenty-fold less MAC operations. In terms of inference speed, the tracing module adds negligible burden to the detection module thanks to our design. The entire model runs at 20fps for the KITTI resolution (370×1224) on an RTX A5000 GPU without careful postprocessing such as quantization, pruning, etc.

IV. Conclusion

This paper contributed ODTFormer, a Transformer-based model to address obstacle detection and tracking problems. We detect the obstacles in the form of a voxel occupancy grid using a deformable attention mechanism and track them by matching voxels across two consecutive frames. We confirmed the effectiveness of ODTFormer by providing extensive quantitative and qualitative results on DrivingStereo and KITTI datasets. Our results showed that ODTFormer achieves state-of-the-art performance on obstacle detection tasks. For the obstacle tracking task, we showed that ODTFormer can achieve accuracy comparable to state-of-the-art methods. The entire model is efficient and runs fast at 20fps

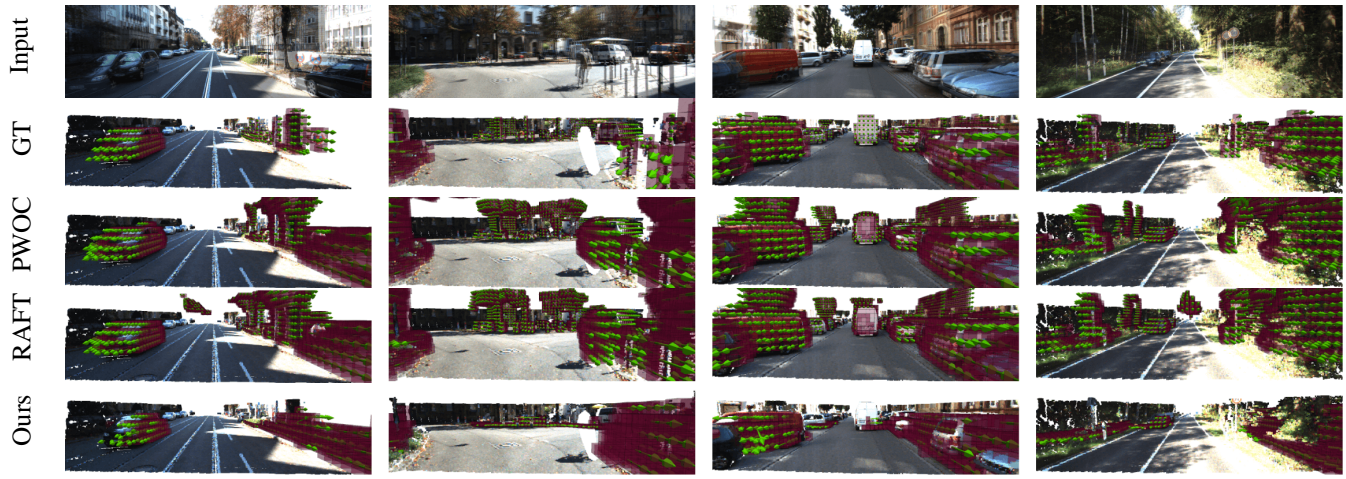


Fig. 3: **Visual results of obstacle tracking on the KITTI 2015 dataset.** The first row shows the stacked images from consecutive frames. The obstacle detection results are shown as red cubes, and the tracking results are marked as green arrows. Longer arrows indicate large motion magnitude.

for the KITTI resolution (370×1224) on an RTX A5000 GPU without careful postprocessing (*e.g.*, quantization, pruning). Being able to detect and track obstacles at the same time will empower a set of downstream tasks, for instance, ensuring safe navigation when people move around a robot.

REFERENCES

[1] J. Frey *et al.*, "Locomotion Policy Guided Traversability Learning using Volumetric Representations of Complex Environments," Aug. 2022, arXiv:2203.15854 [cs].

[2] Y. F. Chen *et al.*, "Socially aware motion planning with deep reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 1343–1350.

[3] N. U. Akmandor *et al.*, "Deep Reinforcement Learning based Robot Navigation in Dynamic Environments using Occupancy Values of Motion Primitives," in *IROS*, 2022.

[4] L. Zhao *et al.*, "E(2)-equivariant graph planning for navigation," *IEEE Robotics and Automation Letters*, vol. 9, no. 4, 2024.

[5] H. Li *et al.*, "Stereonavnet: Learning to navigate using stereo cameras with auxiliary occupancy voxels," Mar. 2024, arXiv:2403.12039 [cs].

[6] Z. Li *et al.*, "BEVFormer: Learning Bird's-Eye-View Representation from Multi-camera Images via Spatiotemporal Transformers," in *Computer Vision – ECCV 2022*, S. Avidan *et al.*, Eds., 2022, pp. 1–18.

[7] Y. Li *et al.*, "VoxFormer: Sparse Voxel Transformer for Camera-Based 3D Semantic Scene Completion," 2023, pp. 9087–9098.

[8] Y. Wei *et al.*, "SurroundOcc: Multi-camera 3D Occupancy Prediction for Autonomous Driving," 2023, pp. 21 729–21 740.

[9] Y. Huang *et al.*, "Tri-Perspective View for Vision-Based 3D Semantic Occupancy Prediction," 2023, pp. 9223–9232.

[10] T. Eppenberger *et al.*, "Leveraging Stereo-Camera Data for Real-Time Dynamic Obstacle Detection and Tracking," in *IROS*, 2020.

[11] Z. Xu *et al.*, "A real-time dynamic obstacle tracking and mapping system for UAV navigation and collision avoidance with an RGB-D camera," in *ICRA*, 2023.

[12] M. Lu *et al.*, "Perception and Avoidance of Multiple Small Fast Moving Objects for Quadrotors With Only Low-Cost RGBD Camera," *IEEE RA-L*, vol. 7, no. 4, Oct. 2022.

[13] Y. Wang *et al.*, "Autonomous Flights in Dynamic Environments with Onboard Vision," in *IROS*, 2021.

[14] H. Li *et al.*, "Stereovoxelnet: Real-time obstacle detection based on occupancy voxels from a stereo camera using deep neural networks," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[15] S. Kareer *et al.*, "ViNL: Visual Navigation and Locomotion Over Obstacles," Jan. 2023, arXiv:2210.14791 [cs].

[16] B. Zhou *et al.*, "RAPTOR: Robust and Perception-aware Trajectory Replanning for Quadrotor Fast Flight," Jul. 2020, arXiv:2007.03465.

[17] A. Loquercio *et al.*, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, p. eabg5810, 2021.

[18] D. Falanga *et al.*, "How Fast Is Too Fast? The Role of Perception Latency in High-Speed Sense and Avoid," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1884–1891, Apr. 2019.

[19] Y. Wang *et al.*, "Pseudo-LiDAR From Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving," 2019, pp. 8445–8453.

[20] A. Vaswani *et al.*, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[21] F. Shamsafar *et al.*, "MobileStereoNet: Towards Lightweight Deep Networks for Stereo Matching," 2022, pp. 2417–2426.

[22] Z. Shen *et al.*, "CFNet: Cascade and Fused Cost Volume for Robust Stereo Matching," 2021, pp. 13 906–13 915.

[23] B. Liu *et al.*, "Local Similarity Pattern and Cost Self-Reassembling for Deep Stereo Matching Networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 2, pp. 1647–1655, Jun. 2022, number: 2.

[24] G. Xu *et al.*, "Attention Concatenation Volume for Accurate and Efficient Stereo Matching," 2022, pp. 12 981–12 990.

[25] X. Zhu *et al.*, "Deformable detr: Deformable transformers for end-to-end object detection," *arXiv preprint arXiv:2010.04159*, 2020.

[26] G. Yang *et al.*, "Drivingstereo: A large-scale dataset for stereo matching in autonomous driving scenarios," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[27] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3061–3070.

[28] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *Proceedings of the 36th International Conference on Machine Learning*. PMLR, May 2019, pp. 6105–6114.

[29] T.-Y. Lin *et al.*, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[30] Y. Xie *et al.*, "Pixel-Aligned Recurrent Queries for Multi-View 3D Object Detection," in *ICCV*, 2023.

[31] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," Sep. 2018.

[32] R. Saxena *et al.*, "Pwoc-3d: Deep occlusion-aware end-to-end scene flow estimation," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 324–331.

[33] H. Jiang *et al.*, "Sense: A shared encoder network for scene-flow estimation," in *ICCV*, 2019.

[34] Z. Teed and J. Deng, "RAFT-3D: Scene Flow using Rigid-Motion Embeddings," Apr. 2021, arXiv:2012.00726 [cs].

[35] N. Mayer *et al.*, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *CVPR*, 2016.