StereoVoxelNet: Real-Time Obstacle Detection Based on Occupancy Voxels from a Stereo Camera Using Deep Neural Networks

Hongyu Li[1,2†], Zhengang Li[3*], Neşet Ünver Akmandor[1,3*], Huaizu Jiang[2], Yanzhi Wang[3], and Taşkın Padır[1,3]

Abstract— Obstacle detection is a safety-critical problem in robot navigation, where stereo matching is a popular vision-based approach. While deep neural networks have shown impressive results in computer vision, most of the previous obstacle detection works only leverage traditional stereo matching techniques to meet the computational constraints for real-time feedback. This paper proposes a computationally efficient method that leverages a deep neural network to detect occupancy from stereo images directly. Instead of learning the point cloud correspondence from the stereo data, our approach extracts the compact obstacle distribution based on volumetric representations. In addition, we prune the computation of safety irrelevant spaces in a coarse-to-fine manner based on octrees generated by the decoder. As a result, we achieve real-time performance on the onboard computer (NVIDIA Jetson TX2). Our approach detects obstacles accurately in the range of 32 meters and achieves better IoU (Intersection over Union) and CD (Chamfer Distance) scores with only 2% of the computation cost of the state-of-the-art stereo model. Furthermore, we validate our method's robustness and real-world feasibility through autonomous navigation experiments with a real robot. Hence, our work contributes toward closing the gap between the stereo-based system in robot perception and state-of-the-art stereo models in computer vision. To counter the scarcity of high-quality real-world indoor stereo datasets, we collect a 1.36 hours stereo dataset with a Jackal robot which is used to fine-tune our model. The dataset, the code, and more visualizations are available at https://lhy.xyz/stereovoxelnet/.

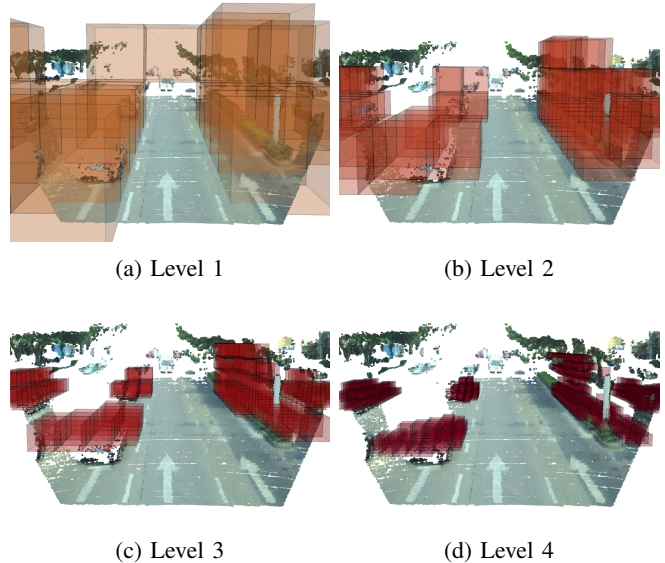(a) Level 1   (b) Level 2

(c) Level 3   (d) Level 4

Fig. 1: Hierarchical output. StereoVoxelNet generates voxels from a stereo pair [5] to represent the detected location of the obstacles in the range of 32 meters in a coarse-to-fine grained manner (from top left to bottom right). The resolutions of the voxel grids are $8^3, 16^3, 32^3$, and 64^3, respectively.

I. INTRODUCTION

To safely navigate around dynamic and unknown environments, robots need to perceive and avoid obstacles in real-time, usually with a limited financial or computation budget. Among all the 3D perception sensors, the stereo camera has been widely used due to its lightweight design and low cost. However, performing depth estimation with the stereo camera is nontrivial where we need to strike a balance between efficacy and efficiency. Therefore, most stereo-based navigation approaches sacrifice the estimation accuracy by using the traditional methods or reducing image resolution to provide real-time feedback [1]–[4].

Deep learning stereo models have shown promising results in depth estimation [6], however, there are mainly two limitations for them to be used in practical and agile navigation. Firstly, even the lightweight deep learning stereo

[1]Institute for Experiential Robotics, Northeastern University, Boston, MA, 02115, USA

[2]Khoury College of Computer Sciences, Northeastern University, Boston, MA, 02115, USA

[3]Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, 02115, USA

*Equally contributed authors

†li.hongyu1@northeastern.edu

model [7] produces inadmissible latency. This is because the dense depth information produced is redundant for obstacle avoidance tasks during motion planning. In most stereo matching procedures, the depth of each pixel is estimated, and most of the individual points are superfluous in close proximity, thus discretized voxels are used to represent the occupancy information in typical robotic applications [1]–[4], [8], [9] that require fast computation.

Secondly, the current state-of-the-art stereo models lack robustness. Even though they can achieve high accuracy in disparity estimation (2D), they are prone to generate more artifacts being transformed in 3D. The artifact is typically around the object boundaries and at a far distance [10], [11]. After the transformation, the artifacts are embodied in the long-tail form [11], due to their pipeline's objective bias. Consequently, an extra outlier filtering is needed to recast the adequate point cloud for robot navigation [4]. Such a procedure often requires careful tuning of parameterization and lacks robustness.

To address the aforementioned issues, we propose an approach that detects the obstacles efficiently and accurately, leveraging the deep learning stereo model. Unlike existing

models that estimate stereo disparity (and then point cloud) from stereo images, our novel end-to-end network directly outputs the occupancy by resorting to a voxel grid, reducing the redundancy of the point cloud. We further improve both the accuracy and the computation cost of the interlacing cost volume [7] by ignoring the obstacles that are far away and avoiding unnecessarily high granularity. Such a design also reduces the noises introduced by the coordinate transformation from the stereo disparity to the 3D point cloud. Our solution can detect obstacles accurately in the range of 32 meters (See Section V) which clearly outperforms the off-the-shelf sensors like Intel RealSense and Microsoft Kinect.

Furthermore, taking advantage of the octree structure and sparse convolution operations, we predict the occupancy of obstacles in a coarse-to-fine grained manner (Fig. 1), preventing waste of computation resources by no further processing the confident unoccupied spaces which are irrelevant to the navigation task at an early stage. Hence, we achieve real-time inference for obstacle detection using an onboard computer. Our approach contributes to closing the gap between computer vision and robot perception.

Over the past decade, numerous autonomous driving stereo datasets have been constructed, such as KITTI [12], DrivingStereo [5], nuScenes [13], and Waymo [14], however, stereo models trained on these datasets typically perform inadequately in the indoor environment. To address the diversity in depth, prior works [7], [15] utilize the synthetic SceneFlow [6] stereo dataset, but encounter performance degradation in the real-world application. Consequently, we construct a stereo dataset containing both indoor and outdoor environments via robot teleoperation and incorporate numerous sensors including IMU, stereo camera, 32-beam LiDAR, etc. More details regarding the dataset are presented in Section IV.

We summarize our contributions as follows:

- We present a novel deep neural network that takes stereo image pairs as input and produces occupancy voxels through an efficient cost volume formulation strategy in real-time using an onboard computer.
- We exploit the sparsity by performing optimization through octree structure and sparse convolution and saving computation resources on confident unoccupied space. We further integrate our perception model with the motion planner module, thus achieving adaptive perception on demand [16]–[18] while navigating.
- We construct a real-world stereo dataset that contains both indoor and outdoor scenes, collected by robot teleoperation. Our contributed dataset complements the existing dataset like JRDB [19] to enhance the dataset scale.

II. RELATED WORK

Obstacle detection using the stereo camera is typically approached by depth estimation, utilizing the classical stereo matching algorithms, such as Semi-global Matching (SGM) [20]. The majority of works construct a pipeline that first estimates the depth map and then extracts the obstacles [1]–[4], [21]. However, such a pipeline lacks efficiency since the depth of every pixel is estimated by the stereo matching module. On the contrary, obstacles typically account for only a small portion of pixels. Pushbroom Stereo [22] tackled the inefficiency by performing block matching sparsely, which is capable of running at 120Hz onboard, but it suffers from high false negatives due to the lack of global attention.

Deep learning end-to-end models are also attempted by several works. [23] developed an end-to-end classification model to perform trail perception and navigation. Similarly [24] and DroNet [25] proposed to use an end-to-end CNN model for the output steering angle. Since this kind of network model learns obstacle avoidance intrinsically, their performance is sensitive to the training scene and they tend to have poor domain adaptation.

Our work in this paper focuses on the sub-modules of visual navigation as in [4], [21] and could improve the overall system by replacing their depth estimation-related modules with our proposed approach.

Stereo matching is used to perform depth estimation by measuring the disparity between stereo images. The classical method like SGM has been widely used in robot perception [1], [2], [4], [21]. As in various computer vision tasks, the performance of the traditional approach has been surpassed by the recent deep learning models and often by a large margin, which also applies to stereo matching tasks [26], [27]. Recent stereo matching work has been focused on developing an end-to-end network to predict the disparity map from a stereo pair [6], [7], [28]. In such an end-to-end network, the matching procedure is often performed by cost volume construction and cost volume aggregation. A cost volume C is used to correlate two unary feature maps generated by the input stereo pair passing through the feature extractor. [7], [29] utilize 3D cost volume formulated as

$$C(d, x, y) = F(f_L(x, y), f_R(x - d, y)), \qquad (1)$$

where F is a function that measures the correlation between the left and right feature maps. d is within the range of $(0, d_{max})$, where d_{max} is a designated maximum disparity level and typically set to 192 in recent works.

DispNet [6] has reached a significantly higher performance compared with SGM on multiple datasets since 2015. Nevertheless, the deployment of recent deep learning stereo models in mobile robotics is still unpopular. In fact, it is a huge challenge to run deep learning stereo models in real-time using an onboard computer.

Octree representation has been widely used in the field of robotics [30], [31]. Our octree decoder design is related to two seminal works: Octree Generating Network [32] and Quadtree Generating Network [18]. While the first one performs 3D reconstruction, the latter work is on 2D image segmentation. Different from these works, we contribute to the adaptation to stereo upstream and its integration with the robot navigation module to achieve adaptive perception on

demand, where the perception granularity can automatically be determined by the robot's demand for obstacle avoidance.
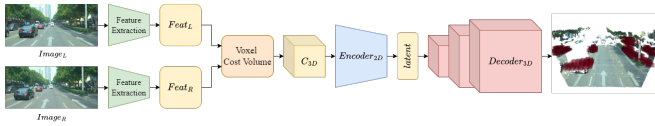
III. STEREOVOXELNET

Fig. 2: Our proposed network, StereoVoxelNet, takes a stereo pair as input and produces a voxel occupancy grid.

Inspired by stereo matching models [7], [26], [28], [33], we design our end-to-end network, StereoVoxelNet, by integrating three main components as in Fig. 2: *i)* feature extraction module for unary feature extraction, *ii)* voxel cost volume: a lightweight cost volume to combine unary features, and *iii)* an encoder-decoder structure for voxels generation. The first two parts extract and aggregate features, while the last part predicts voxels from the low-level feature volume base on an octree structure.

Similar to the navigation framework [8], we define a 3D grid composed of N voxels with respect to the robot's coordinate frame. We denote each voxel as a cubic volume with a side length l_v. The 3D grid $N = n_x \times n_y \times n_z$, where n_x, n_y, and n_z represent the number of voxels on x-axis, y-axis, and z-axis, respectively. Therefore, the physical distance $s_{\{x,y,z\}}$ in each dimension can be calculated as $n_{\{x,y,z\}} \times l_v$. We refer to this volumetric voxel grid structure as our ROI.

Feature Extraction: We utilize a simplified feature extraction module by reducing the number of channels in MobileStereoNet [7]. A feature map of size $C \times \frac{H}{4} \times \frac{W}{4}$ is produced from the feature extraction module. C is the number of feature channels, and H, W is the image's height and width, respectively.

Voxel Cost Volume: We leverage voxel cost volume to aggregate unary features efficiently. A detailed explanation of voxel cost volume is given in Section III-A.

Encoder-Decoder: We use an encoder-decoder component to produce voxels [34]–[37], whereas the encoder is a collection of 2D convolutional layers that extracts a latent vector from the feature volume, and the decoder is composed of a set of 3D deconvolutional layers with an octree generating structure. To predict obstacle occupancy with higher efficiency, we optimize the decoder component by exploiting the sparsity of the occupancy grid and applying sparse convolution. A detailed explanation of our octree optimization is in Section III-B.

A. Voxel Cost Volume

In most recent stereo models, the cost volume is one of the most computation-demanding modules. Furthermore, its output dimension linearly affects the computation cost of the consecutive modules. To improve the efficiency of the entire network, we propose voxel cost based on the interlacing cost volume [7] with a consideration of the physical structure of the final output - voxel occupancy grid. We eliminate

redundant and irrelevant disparity information. Instead of continuously traversing $\{d_i \mid i = 1, 2, \ldots, d_{max}\}$ in equation (1) [7], we replace it with D_{vox}, a subset of disparity levels.

$$C(d, x, y) = Interlace\{f_L(x, y), f_R(x - d_{vox}, y)\}, \quad (2)$$

where $d_{vox} \in D_{vox}$. $Interlace$ is an operator proposed by [7], where the left feature map and right feature map of size $C \times H \times W$ are aggregated one-by-one into $2C \times H \times W$. Since d and z have an inverse relationship, if we traverse continuously on the disparity dimension with a constant gap, we get a skewed distribution of the respective depth z. Particularly, we retrieve sparser features at a farther distance and denser features as getting closer. We use the voxel size l_v as a gap, traversing z in the range of $(0, z_{max})$ where $z_{max} = n_z \times v_{size}$. Consequently, we get $|D_{vox}| = n_z$.

Our algorithm is presented in Algorithm 1. The voxel cost volume comes with two functionalities: *i)* It prunes the information which is not in our ROI. *ii)* It enables a sparser sampling of disparity without harming the overall accuracy.

Algorithm 1 Voxel Cost Volume

Input: f_u, b, z_{max}, α, l_v
Output: C
  $c \leftarrow f_u \times b$ ▷ constant for each stereo camera
  $D_{vox} = \emptyset$
  for $z = 0$ to z_{max} with step size of $\alpha \times l_v$ **do**
    $D_{vox} = D_{vox} \cup \{\frac{c}{z}\}$
  end for
  for d_{vox} in D_{vox} **do**
    $C(d, x, y) \leftarrow Interlace\{f_L(x, y), f_R(x - d_{vox}, y)\}$
  end for

In the algorithm, f_u represents the focal length, b represents the baseline of the stereo camera. α is a constant controlling the traversing step size, and l_v is the size of a voxel as mentioned.

B. Octree Decoder

An octree is a tree-like data structure where each node has eight children [38]. Like binary tree (1D) and quadtree (2D), octree (3D) is a compact representation of the space that advances in both computation efficiency and storage. Benefiting from its tree-like data structure, an octree is naturally hierarchical. We can represent each level of the octree as $l = \{1, 2, \ldots, L_{level}\}$. For the level l, we could obtain the dimension of output $N_l = \delta \times (2^l \times 2^l \times 2^l)$, where δ is the initial resolution. Having a fixed physical size s^3 of the 3D grid, the output of our model follows a coarse-to-fine grained pattern.

Unlike [32] where they define each voxel of output voxel grid y at the location $p \in \mathbb{R}^3$ and octree level l as being one of three states: *empty*, *filled*, or *mixed*, we simplify it into two states *occupied* (1) and *unoccupied* (0), as shown in Fig. 3.

$$y_{l,p} = \begin{cases} 0 \text{ if all the children are unoccupied,} \\ 1 \text{ if any of the children is occupied.} \end{cases}$$
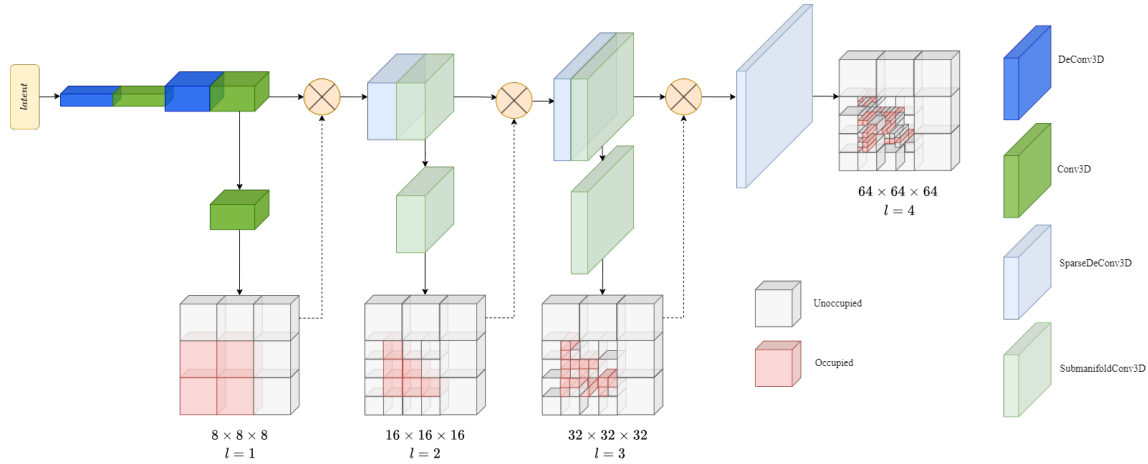
Fig. 3: Our octree decoder takes the latent vector as input and produces coarse-to-fine grained output. The output of each level is used as a mask to prune the feature matrix for the next level. We set the threshold as 0.5 to binarize the mask. Thus, the confident unoccupied spaces become zeros and are pruned using sparse convolutions.

To use a network to predict the occupancy label for each voxel, we feed the latent vector through a set of deconvolution and convolution layers to obtain the first coarse output $y_{1,p}$. We employ occupancy label $y_{1,p}$ as a mask to sparsify the reconstructed feature using element-wise multiplication. The process is repeated until the desired resolution is reached.

To calculate the loss of the octree, we calculate the weighted sum of the negative IoU loss of each tree level.

$$\mathcal{L}_l = \sum_p 1 - IoU(y_{l,p}, \hat{y}_{l,p}),$$

where $y_{l,p}$ and $\hat{y}_{l,p}$ correspond to the prediction and ground-truth label at level l. The overall loss is given by:

$$\mathcal{L} = \sum_l w_l \mathcal{L}_l,$$

where w_l is a designated weight for each level. We set weights w_1 to w_4 as $[0.30, 0.27, 0.23, 0.20]$ empirically.

During inference, we can detect obstacles at the finest level $y_{L_{level},p}$ in real-time utilizing the aforementioned methodologies. However, we further improve the efficiency using a small technique. Leveraging the hierarchical design of the octree decoder, we integrate our StereoVoxelNet with the mobile robot navigation framework and achieve adaptive perception on demand. We use a simple heuristics function to model the octree level variation from the previous observation and permit early exit.

$$l^t = \begin{cases} i - 1 & \text{if } \sum_{p'} y_{i,p'}^{t-1} = 0, \\ i + 1 & \text{otherwise.} \end{cases} \tag{3}$$

where $l^0 = L_{level}$, and we use the superscript t to denote the time step. The heuristic function means if there is no obstacle detected in area p' in front of the robot from the previous timestep $t - 1$, the exit octree level l of current timestep t will decrease by one. In case of an obstacle is detected, the octree level will increase by one.
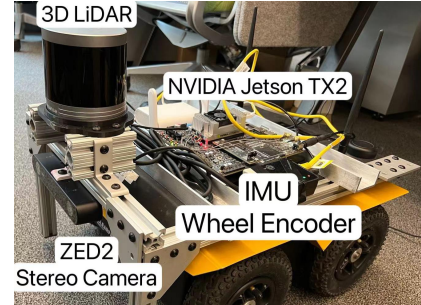
IV. DATASET CONSTRUCTION

Fig. 4: Our Jackal robot is teleoperated in indoor and outdoor environments carrying a suite of sensors to collect data.
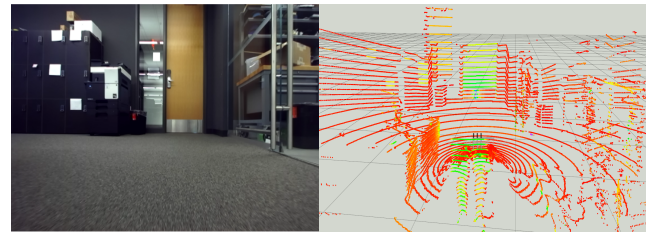

Fig. 5: A sample from our dataset. The left image is rectified image captured by the ZED2 camera. The right image is 3D LiDAR point cloud visualized in RViz.

To collect our dataset, we use a Jackal robot (Fig. 4) from Clearpath Robotics equipped with a front-facing ZED2 stereo camera and a RoboSense-RS-Helio-5515 32-Beam 3D LiDAR which operates at 10Hz. All data is collected, synchronized, and stored in *rosbag* format in ROS [39].

Our human operator controls the robot using a joystick controller. We capture data from different indoor and outdoor scenes within the Northeastern University campus, including different floors in the library and the research facility, the walking bridge, etc. The dataset visualization example is in Fig. 5. Different from social navigation datasets JRDB [19] and SCAND [40] where they mainly operate the robot in a crowded environment and imitate human behavior to avoid obstacles, we intend to operate the robot near the static obstacles while navigating to include more samples of obstacles at close distances.

V. EXPERIMENTS

In this section, we evaluate the performance and efficiency of the proposed obstacle detection solution. We structure our evaluation into two parts: public dataset evaluation and real-world experiment. We also include ablation studies where we examine the effectiveness of our proposed voxel cost volume and octree decoder structure.

We use DrivingStereo [5] to train and evaluate our model. Similar to the widely used KITTI datasets [12], [41], DrivingStereo is a driving dataset but with more samples.

We set $l_v = 0.5$, $n_x = n_y = n_z = 64$. In other words, our ROI is a 64^3 cubic volume in front of us with 32 meters distance, 16 meters left, 16 meters right, and 32 meters towards the top. In practice, we set the size of the latent vector $n_{latent} = 128$. Our model is trained for 30 epochs using an Adam optimizer with a learning rate of 0.001, $\beta_1 = 0.9, \beta_2 = 0.999$. We form a batch size of 16 stereo pairs. To implement the sparse convolution in our octree decoder, we use the *SpConv*[1] library, the submanifold sparse convolution layer [42], [43], and PyTorch [44].

A. Quantitative Results

Similar to 3D reconstruction tasks [34] [45], our final output is a voxel occupancy grid. Therefore, we choose IoU and CD as our evaluation metrics. Besides the accuracy metrics, we also consider computation cost matrices: multiply-accumulate (MACs) and model parameters. Models with more significant MACs typically demand more computation power and process slower.

1) Voxel Cost Volume: We perform an ablation study to evaluate the effectiveness of our proposed voxel cost volume and its impact on the entire network. We compare it with the other three types of cost volume (Table I). The original interlacing cost volume [7] has 48 disparity levels (ICV), and the simplified version of it where we only extract the even number of disparities, results in 24 levels of disparity (Even). We also obtained a further simplified version with only 12 levels of disparity using the same strategy, which has the same disparity levels as our voxel cost volume.

Our voxel cost volume achieves leading performance and close to the least computation cost. Compared with the interlacing cost volume that has the 24 disparity levels, our

Cost Volume Type	CD ↓	IoU ↑	MACs [2] ↓	Parameters ↓
48 Levels (ICV)	3.56	0.32	29.01G	5.49M
24 Levels (Even)	<u>2.96</u>	<u>0.34</u>	21.00G	5.43M
12 Levels (Even)	3.05	0.33	**16.99G**	**5.40M**
12 Levels (**Ours**)	**2.40**	**0.35**	<u>17.31G</u>	**5.40M**

TABLE I: Numerical results on the DrivingStereo testing set. The best result is bold, and the second-best result is underlined.

approach achieves 19% lower CD and 18% lower computation cost. Therefore, we show the effectiveness of cost volume optimization based on ROI.

2) Octree Pruning: In this section, we demonstrate the efficacy of our octree structure pruning strategy by comparing it with the same network using dense grid implementation (Dense). Following the training schemes in [18], we choose the active sites at each octree level using both ground truth (GT) and predicted label (Pred). In addition, we evaluate the baseline network performance that only produces a single output at the end of the network (Straight). Our results are summarized in Table II.

Method	Level	CD ↓	IoU ↑	MACs ↓	Parameters ↓
Straight	4	4.89	**0.38**	17.31G	5.40M
Dense	1	**5.68**	**0.78**	15.05G	5.40M
	2	**3.62**	**0.64**	15.72G	
	3	**2.68**	**0.50**	16.86G	
	4	**2.40**	<u>0.35</u>	17.31G	
Sparse-GT	1	6.75	0.75	14.94G	5.40M
	2	4.16	0.58		
	3	3.05	0.42		
	4	2.56	0.27		
Sparse-Pred	1	<u>5.84</u>	**0.78**	14.94G	5.40M
	2	<u>3.69</u>	**0.64**		
	3	<u>2.75</u>	**0.50**		
	4	<u>2.54</u>	<u>0.35</u>		

TABLE II: Our octree-pruned model (Sparse-Pred) achieves nearly the same performance as the dense grid network while reducing the computation cost.

From the experiment result, we could conclude that the octree decoder achieves near the same performance as the dense grid decoder, while further reducing the computation cost by 13.2%.

3) StereoVoxelNet: We compare the performance of the StereoVoxelNet using voxel cost volume with several state-of-the-art stereo deep learning models and the traditional SGBM algorithm in Table III.

We apply the standard pipeline for obstacle detection using a stereo camera [4], where the disparity estimation is transformed into the depth map, the point cloud, and ultimately the voxel grid.

Our method outperforms the compared approaches by a large margin in terms of accuracy and speed. We show the

---

[1] https://github.com/traveller59/spconv
[2] MACs based on $3 \times 400 \times 880$ RGB stereo pair

[3] Due to CFNet input requirement, MACs based on $3 \times 384 \times 864$
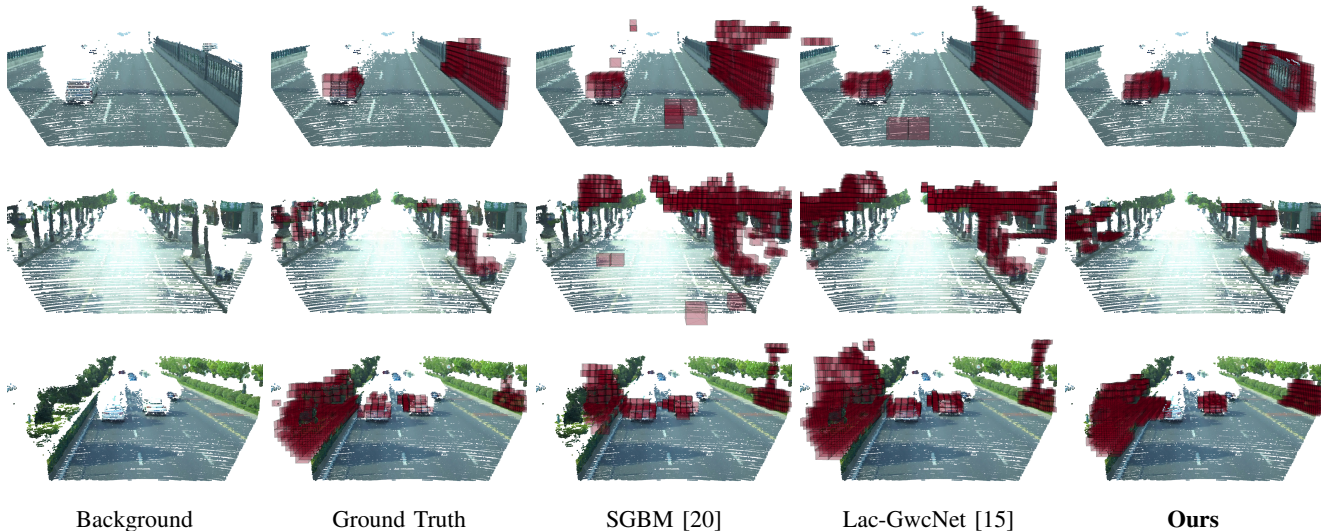[4] Implemented by OpenCV(StereoSGBM)

| Background | Ground Truth | SGBM [20] | Lac-GwcNet [15] | **Ours** |

Fig. 6: Results for DrivingStereo dataset. Output with 64^3 resolution is shown above.

Method	CD ↓	IoU ↑	MACs ↓	Parameters ↓	Runtime ↓
MSNet (2D) [7]	17.03	0.15	91.04G	2.35M	2.61s
MSNet (3D) [7]	12.2	0.21	409.09G	1.77M	
ACVNet [28]	11.36	0.22	644.57G	6.17M	4.38s
Lac-GwcNet [15]	6.22	0.33	775.56G	9.37M	5.83s
CFNet [46] [3]	7.86	0.22	428.40G	22.24M	
SGBM [20] [4]	16.59	0.25	**N/A**	**N/A**	**0.08s**
Ours	**2.40**	**0.35**	17.31G	5.40M	0.55s

TABLE III: Numerical results on the DrivingStereo dataset. We use Intel i7-12700K runtime to compare all approaches.

possibility of applying a deep learning stereo model in the obstacle avoidance task. However, comparing with SGBM in terms of *CPU runtime* is still a huge challenge. Our model runs at **35Hz** on RTX 3090 GPU without any optimization.

B. Qualitative Results

We present the qualitative result in Fig. 6 in which we compare the result with SGBM and the state-of-the-art model Lac-GwcNet [15]. The red voxels represent the detected obstacles. Our method produces the least noise and the closest result to the ground truth. We achieve similar or better results than Lac-GwcNet with only 2% computation cost.
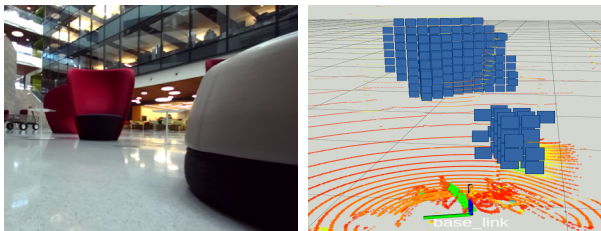
C. Real-World Experiment

Fig. 7: We experiment in an unstructured environment with no prior knowledge. The robot is given a goal across the environment and avoids obstacles during navigation.

We perform real-world autonomous navigation with only a local planner from the *move_base* of ROS Noetic on Ubuntu 20.04. The LiDAR is only used for visualization purposes during the experiment. Our proposed method is the only input to the navigation stack that operates on Jackal's onboard computer (NVIDIA Jetson TX2). The goals are given across the environment (Fig. 7). Although the robot reaches its destination with a high success rate, we observe that noise is often generated by the ground reflection, leading to several failed attempts. Due to the absence of a global planner, the robot is sometimes stuck at local minima. We refer readers to our supplementary video for a vivid demonstration.

VI. CONCLUSION

This paper presented a novel obstacle detection network based on the stereo vision that can run in real-time on an onboard computer. We proposed voxel cost volume to keep the computation within our ROI and match the disparity estimation resolution with the occupancy grid. We leveraged the octree-generating decoder to prune the insignificant computation on irrelevant spaces. Furthermore, we integrated the hierarchical output with the navigation framework to provide adaptive resolution while navigating. To support the data-based robotic applications, we collected and released a high-quality real-world indoor/outdoor stereo dataset via robot teleoperation. Our work shows the possibility of utilizing a deep stereo model in the obstacle detection task.

ACKNOWLEDGMENT

REFERENCES

[1] C. Fu, A. Carrio, and P. Campoy, "Efficient visual odometry and mapping for unmanned aerial vehicle using arm-based stereo vision pre-processing system," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2015, pp. 957–962.

[2] K. Schmid, T. Tomic, F. Ruess, H. Hirschmüller, and M. Suppa, "Stereo vision based indoor/outdoor navigation for flying robots," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 3955–3962.

[3] C. Häne, T. Sattler, and M. Pollefeys, "Obstacle detection for self-driving cars using only monocular cameras and wheel odometry," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 5101–5108.

[4] T. Eppenberger, G. Cesari, M. Dymczyk, R. Siegwart, and R. Dubé, "Leveraging stereo-camera data for real-time dynamic obstacle detection and tracking," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 10 528–10 535.

[5] G. Yang, X. Song, C. Huang, Z. Deng, J. Shi, and B. Zhou, "Driving-stereo: A large-scale dataset for stereo matching in autonomous driving scenarios," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[6] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, arXiv:1512.02134. [Online]. Available: http://lmb.informatik.uni-freiburg.de/Publications/2016/MIFDB16

[7] F. Shamsafar, S. Woerz, R. Rahim, and A. Zell, "Mobilestereonet: Towards lightweight deep networks for stereo matching," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 2417–2426.

[8] N. Ü. Akmandor and T. Padir, "Reactive navigation framework for mobile robots by heuristically evaluated pre-sampled trajectories," *International Journal of Robotic Computing*, vol. 3, no. 1, pp. 47–68, 2021.

[9] N. Ü. Akmandor, H. Li, G. Lvov, E. Dusel, and T. Padır, "Deep reinforcement learning based robot navigation in dynamic environments using occupancy values of motion primitives," *arXiv preprint arXiv:2208.08034*, 2022.

[10] B. H. Wang, C. Diaz-Ruiz, J. Banfi, and M. Campbell, "Detecting and mapping trees in unstructured environments with a stereo camera and pseudo-lidar," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 14 120–14 126.

[11] X. Weng and K. Kitani, "Monocular 3d object detection with pseudo-lidar point cloud," in *IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2019.

[12] A. Geiger, P. Lenz, and R. 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.

[13] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.

[14] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[15] B. Liu, H. Yu, and Y. Long, "Local similarity pattern and cost self-reassembling for deep stereo matching networks," 2021. [Online]. Available: https://arxiv.org/abs/2112.01011

[16] T. L. Dean and M. S. Boddy, "An analysis of time-dependent planning." in *AAAI*, vol. 88, 1988, pp. 49–54.

[17] Z. Liu, Z. Xu, H.-J. Wang, T. Darrell, and E. Shelhamer, "Anytime dense prediction with confidence adaptivity," in *International Conference on Learning Representations*, 2021.

[18] K. Chitta, J. M. Alvarez, and M. Hebert, "Quadtree generating networks: Efficient hierarchical scene parsing with sparse convolutions," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[19] R. Martin-Martin, M. Patel, H. Rezatofighi, A. Shenoi, J. Gwak, E. Frankel, A. Sadeghian, and S. Savarese, "JRDB: A dataset and benchmark of egocentric robot visual perception of humans in built environments," pp. 1–1, conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[20] H. Hirschmuller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, 2005, pp. 807–814 vol. 2.

[21] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, p. eabg5810, 2021. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abg5810

[22] A. Barry and R. Tedrake, "Pushbroom stereo for high-speed navigation in cluttered environments," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015, 07 2014.

[23] A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2016.

[24] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," 2016. [Online]. Available: https://arxiv.org/abs/1604.07316

[25] A. Loquercio, A. I. Maqueda, C. R. del Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.

[26] J. Žbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *J. Mach. Learn. Res.*, vol. 17, no. 1, p. 2287–2318, jan 2016.

[27] A. Tonioni, M. Poggi, S. Mattoccia, and L. Di Stefano, "Unsupervised adaptation for deep stereo," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 1614–1622.

[28] G. Xu, J. Cheng, P. Guo, and X. Yang, "Acvnet: Attention concatenation volume for accurate and efficient stereo matching," *arXiv:2203.02146*, 2022.

[29] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 5695–5703.

[30] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at https://octomap.github.io. [Online]. Available: https://octomap.github.io

[31] M. Labbé and F. Michaud, "RTAB-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," vol. 36, no. 2, pp. 416–446, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21831. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21831

[32] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[33] X. Cheng, Y. Zhong, M. Harandi, Y. Dai, X. Chang, H. Li, T. Drummond, and Z. Ge, "Hierarchical neural architecture search for deep stereo matching," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[34] V. A. Knyaz, V. V. Kniaz, and F. Remondino, "Image-to-voxel model translation with conditional adversarial networks," in *Computer Vision – ECCV 2018 Workshops*, L. Leal-Taixé and S. Roth, Eds. Cham: Springer International Publishing, 2019, pp. 601–618.

[35] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[36] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee, "Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016, p. 1704–1712.

[37] J. Wu, Y. Wang, T. Xue, X. Sun, W. T. Freeman, and J. B. Tenenbaum, "Marrnet: 3d shape reconstruction via 2.5d sketches," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 540–550.

[38] D. Meagher, "Geometric modeling using octree encoding," *Computer graphics and image processing*, vol. 19, no. 2, pp. 129–147, 1982.

[39] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe, Japan, May 2009.

[40] H. Karnan, A. Nair, X. Xiao, G. Warnell, S. Pirk, A. Toshev, J. Hart, J. Biswas, and P. Stone, "Socially CompliAnt navigation dataset (SCAND): A large-scale dataset of demonstrations for social navigation," pp. 1–8, conference Name: IEEE Robotics and Automation Letters.

[41] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3061–3070.

[42] B. Graham and L. van der Maaten, "Submanifold sparse convolutional networks," *arXiv preprint arXiv:1706.01307*, 2017.

[43] B. Graham, M. Engelcke, and L. Van Der Maaten, "3d semantic segmentation with submanifold sparse convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9224–9232.

[44] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[45] S. Kong, F. Shi, C. Wang, and C. Xu, "Point cloud generation from multiple angles of voxel grids," *IEEE Access*, vol. 7, pp. 160 436–160 448, 2019.

[46] Z. Shen, Y. Dai, and Z. Rao, "Cfnet: Cascade and fused cost volume for robust stereo matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 13 906–13 915.