Cross-view Semantic Segmentation for Sensing Surroundings

Bowen Pan[1], Jiankai Sun[2], Ho Yin Tiga Leung[2], Alex Andonian[1], Bolei Zhou[2]

***Abstract*— Sensing surroundings plays a crucial role in human spatial perception, as it extracts the spatial configuration of objects as well as the free space from the observations. To facilitate the robot perception with such a surrounding sensing capability, we introduce a novel visual task called Cross-view Semantic Segmentation as well as a framework named View Parsing Network (VPN) to address it. In the cross-view semantic segmentation task, the agent is trained to parse the first-view observations into a top-down-view semantic map indicating the spatial location of all the objects at pixel-level. The main issue of this task is that we lack the real-world annotations of top-down-view data. To mitigate this, we train the VPN in 3D graphics environment and utilize the domain adaptation technique to transfer it to handle real-world data. We evaluate our VPN on both synthetic and real-world agents. The experimental results show that our model can effectively make use of the information from different views and multi-modalities to understanding spatial information. Our further experiment on a LoCoBot robot shows that our model enables the surrounding sensing capability from 2D image input.**

I. INTRODUCTION

Recent progress in semantic understanding enables machine perception to segment a scene precisely into meaningful regions and objects [1], [2]. These semantic segmentation techniques have benefited many automation applications, like autonomous driving [3]. Though the semantic segmentation network can recognize semantic content in a static image, it is still far from enough to facilitate robots to sense in an unknown environment and navigate freely there. One important reason is that the parsed first-view semantic mask is still at pure image-level without providing any spatial information about the surroundings. Many attempts have been made to extract the spatial knowledge from image input [4], [5]. However, most of them try to make robots perceive the environment only through obtaining their current location instead of having a deeper semantic understanding of the surrounding objects. To perceive spatial configuration from pure image input, an intuitive approach is to explicitly train networks to infer the top-down-view semantic map which directly contains the spatial configuration information of the surrounding environment. Based on the top-down-view semantic map we can then infer the position coordinates and functional properties of surrounding regions and objects.

To enable machines to capture the spatial structure of the surroundings from 2D images, we explore a new image-based scene understanding task, *Cross-View Semantic Segmentation*. Different from the standard semantic segmentation predicting the labels of each pixel in the input image, the cross-view semantic segmentation aims at predicting the top-down-view semantic map from a set of first-view observations (see Fig. 1). The resulting top-down-view semantic map, as a 2.5D spatial representation of the surrounding, indicates the spatial layout of the discrete objects such as chair and human, as well as the stuff classes floor and wall. Notice that although there is a huge literature of 3D methods to reconstruct environments [6] our method has its unique advantages. For example, robot perception systems based on 3D sensors involve expensive cost not only in sensor setup but also in computational power. Instead, the top-down-view map from the cross-view semantic segmentation can facilitate the robot to understand its surroundings in a lightweight and efficient way. In many situations such as free space exploration for mobile robots where the height information is not that essential, the 2D top-down-view semantic map would be sufficient to provide spatial information with much less computation cost.

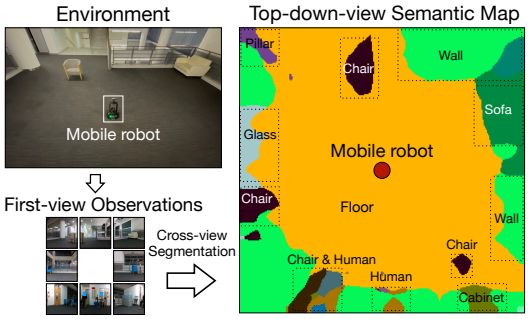

Fig. 1: Top-down-view semantics is predicted from the first-view real-world observations in the cross-view semantic segmentation. Input observations from multiple angles are fused. Notice that the result in this figure is generated without training on real-world data.

One challenge in cross-view semantic segmentation is the difficulty of collecting the top-down-view semantic annotations. Recently, simulation environments such as House3D [7] and CARLA [8] have been proposed for training navigation agents. In these environments, cameras can be placed at any location in the simulated scene while the observations in multiple modalities can be extracted. Here the modalities refer to the different formats of data such as RGB images, semantic masks, and depth maps. Thus, we leverage the simulation environments to acquire cross-view annotated data. To reduce the domain gap between the synthetic scenes and the real-world scenes, we transfer the models trained in the simulation environment to the real-world scenes through domain adaptation. Furthermore, to validate the robustness in the real-world scene, we use a mobile robot to predict the

[1]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, [2]Department of Information Engineering, The Chinese University of Hong Kong

top-down-view semantic map in the real indoor environment and then chase the target.

In this work, we propose a novel framework with View Parsing Network (VPN) for cross-view semantic segmentation using simulation environments and then transfer them to real-world environments. In VPN, a view transformer module is designed to aggregate the information from multiple first-view observations with different angles and different modalities. It further outputs the top-down-view semantic map with a spatial layout of objects. We evaluate the proposed models on the indoor scene of the House3D environment [7] and the outdoor driving scene of the CARLA environment [8]. Experiments show that, on synthetic data, our model achieves 85.0% pixel accuracy and 41.0% mIoU in House3D as well as 84.7% pixel accuracy and 33.2% mIoU in CARLA, and on realistic data, we get the performance of 78.8% pixel accuracy on nuScenes [9], a real-world dataset for autonomous driving, showing the potential of our work for parsing real-world data. Furthermore, to show the cross-view semantic task helps visual navigation, we have demonstrations of simulated agents and real robot respectively. More details are revealed in Section IV,V,VI and the attached video.

Our main contributions are as follows: (1) We introduce a novel task named *cross-view semantic segmentation* to facilitate robots to flexibly sense the surrounding environment. (2) We propose a framework with *View Parsing Network* which effectively learns and aggregates features across first-view observations with multiple angles and modalities. (3) We further apply the domain adaptation technique to transfer our model to work in real-world data while without any extra annotations.

II. RELATED WORK

A. Semantic Segmentation and Semantic Mapping

The semantic segmentation task generates a pixel-wise semantic map to label each pixel of the input image. Deep learning networks for semantic segmentation [10] are designed to segment the image pixel-wise within one-view. Image datasets with pixel-wise annotations such as CityScapes [3] are used for the training of semantic segmentation networks. There is also a huge literature about semantic mapping in robotics domain [1], [2], [4], [5], which provides a semantic abstraction of the environment and a way to communicate with robots.

B. Layout estimation and view synthesis

Estimating layout has been an active topic of research (i.e. room layout estimation [11], free space estimation [12], and road layout estimation [13], [14]). Most of the previous methods use annotations of the layout or geometric constraints for the estimation, while our proposed framework estimates the top-down-view map directly from the image, without the intermediate step of estimating the 3D structure of the scene. On the other hand, view synthesis has been explored in many works [15], [16]. They focus on generating realistic cross-view images while cross-view segmentation aims at parsing semantics across different views.

C. Learning in Simulation Environments

Given that current graphics simulation engines can render realistic scenes, recognition algorithms can be trained on data pulled from simulation engines (i.e. for visual navigation models [17]). Several techniques have been proposed to address the domain adaptation issue when models trained with simulated images are transferred to real scenes [18]. Rather than working on the task of visual navigation directly, our work aims at parsing the top-down-view semantic map from the first-view observations. The resulting top-down-view map will further facilitate visual navigation and path planning.

III. CROSS-VIEW SEMANTIC SEGMENTATION

A. Problem Formulation

The objective of cross-view semantic segmentation is as follows: given the first-view observations as input, the algorithm must generate the top-down-view semantic map. The top-down-view semantic map is a map captured by a camera at a certain height from the top-down view with the annotations of the semantic label of each pixel. The input first-view observations are a set of images with different modalities. They are captured at N different angles by the robot's camera (with $360/N$ degrees apart).

B. Framework of the View Parsing Network

Fig. 2 illustrates two stages of our framework. In the first stage, we propose View Parsing Network (VPN) to learn and aggregate features from multiple first-view observations in the simulation environment. In VPN, first-view observations are first fed into the encoder to extract first-view feature maps. For each modality, VPN has a corresponding encoder to process it. All of these first-view feature maps from different angles and different modalities are transformed and then aggregated into one top-down-view feature map in the View Transformer Module. Then the aggregated feature map is decoded into a top-down-view semantic map. Details of how to transform and aggregate these first-view feature maps can be found in Sec. III-D. In the second stage of our framework, we transfer the knowledge which VPN learns from the simulation environment to the real-world data. We slightly modified the domain adaptation algorithm proposed by [18] to fit our cross-view semantic segmentation task and our VPN architecture. More details of this part will be revealed in Section III-C.

Pipeline. As shown in Fig. 2, from one spatial position in a 3D environment, we first sample $N \times M$ first-view observations from N angles and M modalities (here $N = 6, M = 2$ in Fig. 2) in even angles so that all-around information is captured. The first-view observations are encoded by M encoders for M corresponding modalities respectively. These CNN-based encoders extract $N \times M$ spatial feature maps for their first-view input. Then all of these feature maps are fed into the View Transformer Module (VTM). VTM transforms these view feature maps from first-view space into the top-down-view feature space and fuses them to get one final feature map which already contains sufficient spatial

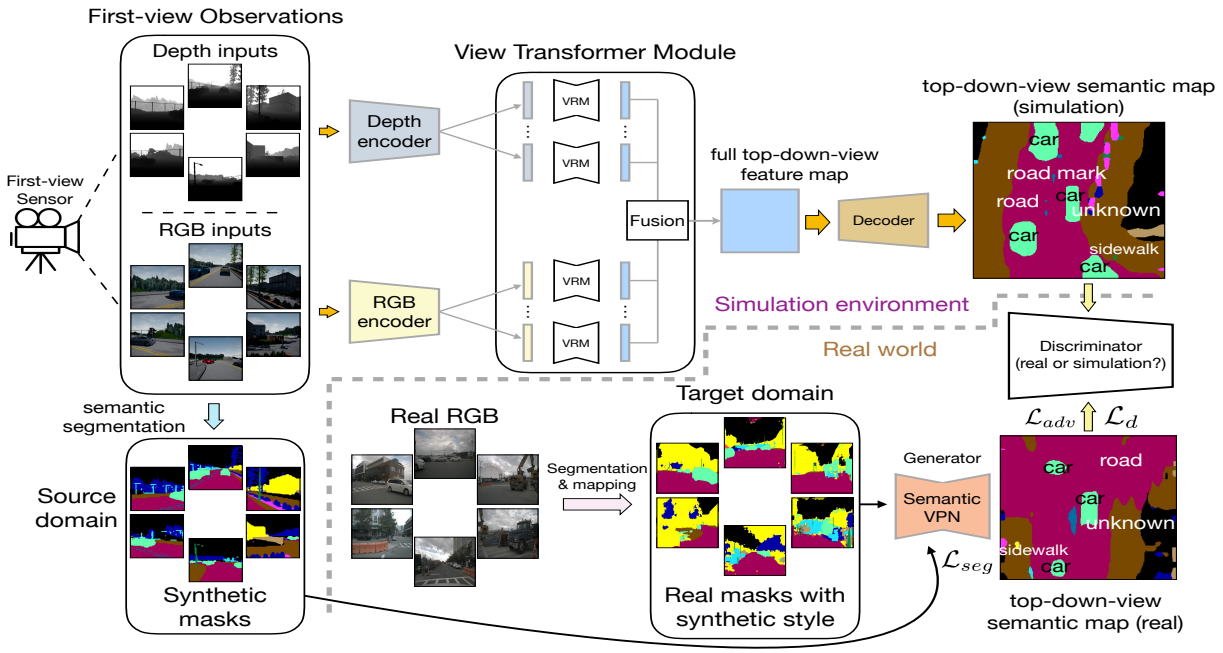

Fig. 2: Framework of the View Parsing Network for cross-view semantic segmentation. The simulation part illustrates the architecture and training scheme of our VPN, while the real-world part demonstrates our domain adaptation process for transferring our VPN to the real world.

information. Finally, we decode it to predict the top-down-view semantic map using a convolutional decoder.

View Transformer Module. Although the encoder-decoder structure gets huge success in the classical semantic segmentation area [10], our experiment (cf. Table III) shows that it performs poorly in the cross-view semantic segmentation task. We conjecture that it is because in standard semantic segmentation architecture the receptive field of the output spatial feature map is roughly aligned with the input spatial feature map. However, in cross-view semantic segmentation, each pixel on the top-down-view map should consider all input first-view feature maps, not just at a local receptive field region.

After thinking about the flaws of the current semantic segmentation structure, we design the View Transformer Module (VTM) to learn the dependencies across all the spatial locations between the first-view feature map and the top-down-view feature map. VTM will not change the shape of input feature map, so it can be plugged into any existing encoder-decoder type of network architecture for classical semantic segmentation. It consists of two parts: View Relation Module (VRM) and View Fusion Module (VFM). The diagram at the central of Figure 2 illustrates the whole process: The first-view feature map is first flattened while the channel dimension remains unchanged. Then we use a view relation module R to learn the relations between the any two pixel positions in flattened first-view feature map and flattened top-down-view feature map. That is:

$$f_t[i] = R_i(f[1], ..., f[j], ..., f[HW]), \tag{1}$$

where $i, j \in [0, HW)$ are the indices of top-down-view feature map $t \in R^{HW \times C}$ and first-view feature map $f \in R^{HW \times C}$ respectively along the flattened dimension, and R_i models the relations between the i^{th} pixel on top-down-view feature map and every pixel on first-view feature map. Here we simply use multilayer perceptron (MLP) in our view relation module R. After that, the top-down-view feature map is reshaped back to $H \times W \times C$. Notice that each first-view input has its own VRM to get the top-down-view feature map $t^i \in R^{H \times W \times C}$ based on its own observations. To aggregate the information from all observation inputs, we fuse these top-down-view feature map t^i by using VFM. More details of VFM and VRM will be introduced in Sec. III-D.

C. Sim-to-real Adaptation

To generalize our VPN to real-world data without the real-world ground truth, we implement the sim-to-real domain adaptation scheme shown in Fig. 2 to narrow the gap. This scheme contains the following pixel-level adaptation and output space adaptation.

Pixel-level adaptation. To mitigate the domain shift, we adopt the pixel-level adaptation on the real-world inputs to make them look more like the style of the simulation data. Considering that it is still difficult to perfectly transfer the real RGB pixels to the simulation space, we transform the RGB data in both the real world and simulation environment to semantic masks. Semantic mask is an ideal mid-level representation without texture gap while including sufficient information and it is easy to transfer. In the simulation environment, semantic masks can be easily generated and we can use them to learn to segment RGB images. As for real-world data, we use the existing segmentation model to extract semantic masks from RGB images and map the categories to match the categories in the simulation environment. This

TABLE I: Results on House3D cross-view dataset with different modalities and view numbers.

	3D Geometric Baseline		X-Fork(RGB) in [16]		RGB VPN		Semantic VPN		Depth VPN	
Networks	PA	mIoU	PA	mIoU	PA	mIoU	PA	mIoU	PA	mIoU
1-view model	31.3%	2.4%	38.0%	1.5%	55.8%	6.5%	59.6%	13.2%	56.9%	7.6%
2-view model	46.8%	7.2%	40.0%	1.9%	70.1%	14.8%	75.7%	25.9%	70.2%	15.6%
4-view model	63.2%	22.8%	39.5%	2.0%	80.3%	27.2%	**85.0**%	40.6%	77.3%	22.0%
8-view model	67.6%	27.1%	43.9%	1.8%	**81.2**%	**28.5**%	84.7%	**41.0**%	**82.1**%	**29.9**%

TABLE II: Results of cross-modality learning for VPN. Here we compare the results with the inputs from 4 views.

Method	Pixel Accuracy	mIoU
RGB VPN	80.3%	27.2%
Depth VPN	77.3%	22.0%
Semantic VPN	85.0%	40.6%
R+D (late fusion)	81.2%	27.3%
R-D VPN	82.8%	31.2%
D+S (late fusion)	83.5%	33.9%
D-S VPN	**86.3**%	43.2%
S+R (late fusion)	84.3%	35.7%
S-R VPN	85.1%	42.3%
D+S+R (late fusion)	84.3%	29.4%
D-S-R VPN	86.2%	**43.6**%

process can be formulated as follows:

$$\{I_S\} = \mathscr{M}_{Real \to Synthetic}(\mathscr{P}_{RGB \to Mask}(\{I_R\})), \qquad (2)$$

where I_R, I_S are the real RGB image and synthetic-style semantic mask respectively, $\mathscr{P}_{RGB \to Mask}$ is the existing semantic segmentation model which parses the real-world RGB into semantic mask, and $\mathscr{M}_{Real \to Synthetic}$ is the semantic category mapping process where we construct the concept mappings between the real world and the simulation environment, for instance, the "human" category in real world and the "pedestrain" category in simulation environment.

Output space adaptation. Beyond the pixel-level transfer on input data, we also devise an adversarial training scheme in structured output space based on the method proposed in [18]. Here the generator $\mathscr{G}$ is a view parsing network generating the top-down-view prediction P, and the discriminator $\mathscr{D}$ is used to discriminate if P is generated from the source domain. The generator $\mathscr{G}$ is initialized by the weights of a VPN trained on the semantic data in the simulation environment as we illustrated before. During the training phase, we first forward a group of input images from the source domain $\{I_s\}$ to $\mathscr{G}$ and optimize it with a normal segmentation loss $\mathscr{L}_{seg}$. Then we use $\mathscr{G}$ to extract the feature map F_i (after the softmax layer) of the images from the target domain $\{I_t\}$ and use discriminator to distinguish whether F_t is from the source domain. Notice that $\{I_s\}$ is a cluster of synthetic semantic masks in the simulation environment and $\{I_t\}$ contains the real-world mask with synthetic style. Here $\mathscr{G}$ is updated by the gradients propagated from $\mathscr{D}$, while the weights of $\mathscr{D}$ is fixed, which encourages $\mathscr{G}$ to unify the feature distributions of the source domain and target domain. Lastly, we optimize the discriminator by training $\mathscr{D}$ to recognize which domain the feature output is from. The loss function to optimize $\mathscr{G}$ can be written as follows:

$$\mathscr{L}(\{I_s\},\{I_t\}) = \mathscr{L}_{seg}(\{I_s\}) + \lambda_{adv}\mathscr{L}_{adv}(\{I_t\}), \qquad (3)$$

where $\mathscr{L}_{seg}$ is the cross-entropy loss for semantic segmentation, $\mathscr{L}_{adv}$ is designed to train the $\mathscr{G}$ and fool the discriminator $\mathscr{D}$. The loss function for the discriminator $\mathscr{L}_d$ is a cross-entropy loss for binary source & target classification.

D. Network configuration

View encoder and decoder. To balance efficiency and performance, we use ResNet-18 as the encoder. We remove the last Residual Block and the Average Pool layer so that the resolution of the encoding feature map remains large, which better preserves the details of the view. We employ the pyramid pooling module used in the standard scene parsing [10] as the decoder.

View Transformer Module. For each view relation module, we simply use the two-layer MLP. We choose this because two-layer MLP doesn't bring too much extra computation so that we can keep our model following the lightweight-and-efficient rationale. Input and output dimensions of the VRM are both $H_I W_I$, where H_I and W_I are respectively the height and width of the intermediate feature map. We flatten the intermediate feature map to $C_I \times W_I H_I$ before we input and reshape it back to $C_I \times W_I \times H_I$ after that. As for the view fusion module, we just add all the features up to keep the shape consistent. Notice that we could come up with more sophisticated designs for VRM and VFM, but currently we focus more on streamlining the whole pipeline of the cross-view semantic segmentation.

Sim-to-real. For the generator $\mathscr{G}$, we use the architecture of the 4-view VPN. For the discriminator $\mathscr{D}$, we adopt the same architecture in [18]. It has 5 convolution layers, each of which is followed by a leaky ReLU with the parameter 0.2 (except the last layer). We use HRNet [19] pretrained on CityScapes dataset [3] to extract the semantic mask from real-world images.

IV. EXPERIMENTS

We first go through the overview of the cross-view segmentation datasets in Section IV-A. Then we show the performance of VPN on synthetic data of the House3D and

TABLE III: Ablation study of View Transformer Module.

Modality	VPN w/o VTM		VPN	
1-view	Pix. Acc.	mIoU	Pix. Acc.	mIoU
RGB	53.9%	6.3%	**55.8**%	**6.5**%
Depth	55.7%	6.5%	**56.9**%	**7.6**%
Semantic	57.4%	10.0%	**59.6**%	**13.2**%
8-view	Pix. Acc.	mIoU	Pix. Acc.	mIoU
RGB	60.5%	8.7%	**81.2**%	**28.5**%
Depth	43.8%	2.5%	**82.1**%	**29.9**%
Semantic	47.6%	6.5%	**84.7**%	**41.0**%

CARLA environment in Section IV-B. Finally in Section IV-C, we demonstrate the real-world performance of our VPN which is trained in the simulation environment.

A. Benchmarks

Here we introduce *two synthetic cross-view datasets*, *House3D cross-view dataset* and *Carla cross-view dataset*, and *one real-world cross-view dataset*, *nuScenes dataset*.

House3D cross-view dataset. We build a synthetic indoor-room dataset from the House3D environment [7]. In our experiments, we select 410 scenes in House3D to construct a dataset with cross-view data annotations. We refer to this dataset as *House3D Cross-view Dataset*. Each data pair contains 8 first-view input images captured from 8 different orientations with 45 degrees apart. Additionally, each data pair comes with the top-down-view semantic mask captured in the ceiling-level height. To be complete, we store the input image with multiple modalities including the RGB images, depth maps, and semantic masks. In each scene, we sample the data with a 0.5-meter stride over the floor map. We split the dataset into training and validation set. The training set contains 143k data pairs from 342 scenes while the validation set contains 20k data pairs from 68 scenes.

NuScenes dataset. NuScenes is a public large-scale dataset for autonomous driving which contains 1000 driving scenes in Boston and Singapore. Each scene has a 20-second-length driving clip consists of multiple data samples. And each data sample contains first-view RGB images from 6 directions (*Front, Front-right, Back-right, Back, Back-left, Front-left*) in different modalities. NuScenes also provide the map extension of the whole scene. We utilize this map to extract the local top-down view map for each data sample. However, categories on the map extension focus more on the functional properties of each part of the road, such as lane and parking lot, which do not exactly match our semantic model trained in the simulation environment. To effectively use the map information, we only extract the local top-down-view map with only drivable area information to quantitatively evaluate our VPN in real world. We select 919 data samples without the top-down-view mask for unsupervised training and 515 data samples with the binary top-down-view mask for evaluation.

CARLA cross-view dataset. To match the data composition in NuScenes, we use CARLA simulator to extract data. CARLA is a popular open-source simulator for training and evaluation of autonomous driving. To build the synthetic source domain dataset, we extract $28,000$ data pairs with top-down-view annotations and different input modalities from 14 driving episodes. Each data pair contains 6 first-view input image sets captured from the same 6 directions.

B. Evaluation

We present VPN performances on the synthetic data of House3D cross-view and CARLA cross-view datasets.

Metrics. We report the results of cross-view semantic segmentation using two commonly used metrics in semantic segmentation: PIXEL ACCURACY (PA) which characterizes the proportion of correctly classified pixels, and MEAN IOU (MIOU) which indicates the intersection-and-union between the predicted and ground truth pixels.

Baselines. Since there are no previous work experiments on cross-view segmentation task, two methods are included as the comparison baselines: (1) *3D geometric method*. With the observed depth and RGB images, we can reconstruct the 3D points cloud with the voxel-level semantic label. For each pixel on the depth image, we back-project it from pixel coordinates to the 3D coordinates in the world frame by using the camera's intrinsic and extrinsic matrices. We then attach its semantic label from the 2D semantic mask image to the voxel. After that, we reproject 3D semantic points cloud into the top-down view. Finally, we use a small network which learns to fill the holes on the obtained top-down view to generate the final top-down-view semantic mask. (2) *Cross-view synthesis.* We also compare with the architecture which is used in cross-view image synthesis literature [16]. In [16], Regmi *et. al.* use a conditional GAN called X-Fork to generate aerial images from street-view images. We use the encoder in X-Fork to encode the first-view RGB images and add them up. After that we use the decoder in X-Fork to generate the top-down-view semantic map.

1) Results of VPNs: We present the results of our VPN for cross-view semantic segmentation in House3D, including the ones of single-modality and multi-modalities VPN respectively. To better evaluate our VPN, we impose an upper bound that we perform segmentation using top-view RGB images directly as inputs, where we get the performance of 91.4% *pixel acc.* and 41.2% *mIoU*. We also show the comparison with the geometric baseline and the ablation study of View Transformer Module.

Single-modality VPN. We show the House3D results of single-modality VPN with different modalities and different numbers of views in Table I. We can see that as VPN receives more views, the segmentation results improve rapidly. We also plot some qualitative results by our VPNs in Fig. 3. As for Carla dataset, we achieve the performance of *84.7% pixel acc.* and *33.2% mIoU* with training a 6-view RGB-input model.

Multi-modalities VPN. We demonstrate the results of multi-modalities VPN in Table II to show that our VPN can effectively synthesize information from multiple modalities.

We set the late-fusion baseline to compare with our multi-modalities VPN, which simply averages the softmax outputs of each single-modality VPN to obtain the final results. We find that the Depth-Semantic VPN achieves the best performance and makes a great improvement. This may be because semantic mask and depth map are two complementary information. However, the Semantic-RGB combination does not bring too much improvement. The reason can be that, for this cross-view semantic segmentation task, semantic input contains most of the useful information in the RGB.

Importance of View Transformer Module. We further evaluate our model in Table III to show the importance of the view transformer module. The baseline network is a classic encoder-decoder architecture used in the standard semantic segmentation, in which the encoder and the decoder are the same as our VPN. We train the baseline model with input view numbers as 1 and 8 respectively. We simply sum up the feature maps from different views and then feed it to the decoder. Our VPN easily outperforms the baseline and, in some multi-view cases, the baseline model does even worse than single-view one due to the bad fusion strategy.

Comparing with baseline. Table I shows that our VPN can easily outperform the 3D geometric method. 3D Geometric method is very easy to fail when there are obstacles. It tends to integrate information from multiple steps to understand the spatial structure of an entire room. In Fig. 3, we can see that the 3D geometric method is unable to reconstruct the objects which can not be directly observed, even after filling the holes, such as the desk behind the chairs shown in the figure. As for X-Fork, we can see that the original generator performs badly in our cross-view semantic segmentation task. This is because X-Fork doesn't have a necessary module to transform the first-view feature map into the top-down-view space. The ablation study in Table III shows a similar issue that there is a significant performance drop when VPN doesn't contain the VTM.

C. Results of sim-to-real adaptation

After we train and test our VPNs in the simulation environment, we transfer our model to the real-world data. Notice that such a process is non-trivial since there is a huge gap between image appearance distributions in real world and CARLA simulator. We first train a 6-view semantic VPN model on the predicted semantic masks in CARLA simulator and then transfer it to nuScenes dataset by using an unsupervised domain adaptation process as depicted in Section III-C. We provide the qualitative results in Fig. 4, from which we can see that our VPN can roughly segment various road shapes like crossroads and also sketch the relative locations of surrounding objects such as cars and buildings. As shown in Table IV, we also evaluate the quantitative results of real-world performance by using binary drivable-area ground truth.

TABLE IV: Results in real world.

Method	Pix. Acc.	Mean Acc.	mIoU
Before Adaptation	72.6%	61.4%	28.0%
After Adaptation	**78.8%**	**65.2%**	**31.9%**

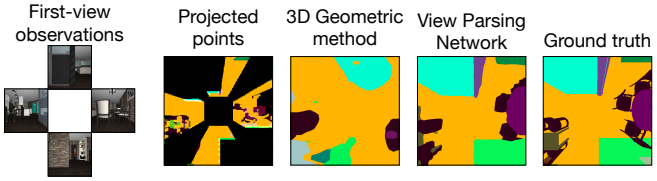

Fig. 3: Qualitative results of 3D geometric method and our VPN. Considering that the geometric method requires semantic mask and depth map, we use the 4-view Depth-Semantic VPN to predict the top-down-view semantic map to fairly compare these two methods.

V. EXPLORATION WITH TOP-DOWN-VIEW MAP

When exploring an unknown space our humans head to the regions which they have not visited. This intuition reflects that exploration requires the agent to identify free space as well as remember which areas it has not visited yet. To achieve this goal, we make the agent able to identify the free space by training it to predict the top-down-view free-space map. Along with the state map which records the already-executed action sequence, the agent can remember the unvisited area.

Top-down-view free-space map. We train the VPN to predict top-down-view free-space map. Different from the semantic map, free-space map has only two categories, obstacle and free space, which are denoted by 0 1 respectively.

State map. Due to the ideal assumption made above, by memorizing the previous actions it has executed, the agent can easily build the state map which contains the information of the already-visited positions. We label the unvisited pixels as 0 and the already-visited pixels as 1 on the state map.

Exploration algorithm. We detail the navigation policy decision algorithm in Algorithm 1. At each time step t, we make the action a_t and update the agent with the next top-down-view free-space map T_{t+1} and state map S_{t+1}. In both the top-down-view free-space map and the state map, we assume that the agent is always at the center of the map.

A. Result and comparison

To demonstrate that VPN can help navigation, we compare it with the following baselines for exploration. *Random walk:* Random walk agent randomly chooses one action from Forward, Back, Right-forward and Left-forward, at each time step. *Top-down-view navigation with ground truth (GT):* By planning on the ground truth top-down-view free-space map with Algorithm 1, we can obtain the upper-bound performance of our method. The difference is that in our case the top-down-view free-space map is predicted by VPN, rather than the ground truth. *Imitation learning (IL) without top-down-view:* A reactive CNN network learns to imitate the expert exploration trajectories given the first-view observations. The trajectories are generated by the baseline above with a top-down-view ground truth map. Network inputs are 4 first-view depth images. We also input the state map to indicate the already-visited area. We extract 729

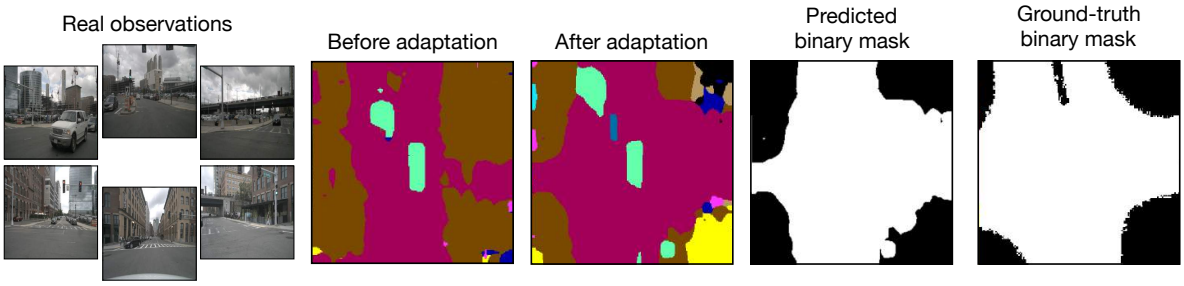

Fig. 4: Qualitative results of sim-to-real adaptation. We provide the results of source prediction both before and after domain adaptation, drivable area prediciton after adaptation and the groud-truth drivable area map.

Algorithm 1 Exploration decision policy at time t

Input: A top-down-view free-space map T_t and a state map S_t at time step t, where $T_t, S_t \in \{0,1\}^{L\times L}$.

Output: Policy action a_t, where $a_t \in$ {Forward, Back, Left-forward, Right-forward, Done}.

$U_t \leftarrow T_t \bigcap \neg S_t$; $a_t \leftarrow$ Done; $d_s \leftarrow +\infty$
$D_t \leftarrow$ computeDistMap(U_t)
for a in {Forward, Back, Left-forward, Right-forward} **do**
    $d =$execute(a)
    if $d_s > d$ **then**
        $d_s \leftarrow d$; $a_t \leftarrow a$
    end if
end for
return a_t

computeDistMap(): Compute the shortest distance of each map pixel to the unvisited free-space region.
execute(): Return the shortest distance of the pixel to which the agent transit if execute the action a.

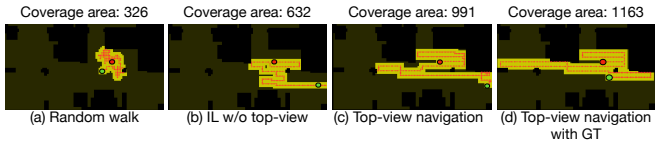

Fig. 5: Examples of the surrounding exploration. Each method has the same start point. The trajectory is shown and the explored area is lighted up. Start point and end point are marked as red point and green point respectively

trajectories for the training set and 121 trajectories for the validation set to train the navigation agent. Each trajectory contains 150 states which are all labeled with expert policy.

TABLE V: Comparison on exploration.

Method	Coverage Area
Random walk	260.3 ± 82.7
IL w/o top-down-view	443.8 ± 340.6
Top-down-view navigation	673.8 ± 349.8
Top-down-view navigation with GT	1070.8±326.2

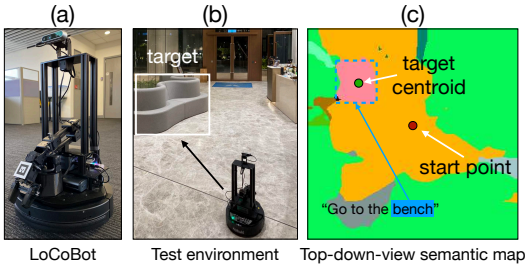

Fig. 6: Our experiments are conducted on a LoCoBot mobile robot in the PyRobot platform. (a) We show a picture of the LoCoBot robot. (b) We show our test environment and the target specified by a semantic token (e.g. bench). (c) We show the process that parses the instruction and calculate the target coordinates.

We run the algorithm directly on our predicted top-down-view map. For testing all the methods, we start the episode by initializing the state maps from zero, indicating that all free space is yet to be visited. *Coverage Area* is defined to measure exploration performance. We randomly choose 100 starting points on a scene map. For each starting point, we let the agent explore the space for 300 steps and compute the coverage area. Then final results are obtained by averaging the coverage area of these 100 episodes. Table V plots the exploration result for different methods and Fig. 5 shows some sample trajectories. We can see that equipped with the predicted top-down-view map from our VPN, the agent can efficiently explore the environment.

VI. REAL ROBOT EXPERIMENT

To verify the performance of our model in the real-world robotic environment, we conduct a semantic navigation experiment by using a LoCoBot mobile robot [20] (cf. Fig. 6a). In this task, the robot is required to identify and reach the target specified by a semantic token. For instance, given the instruction "go to the bench", the robot has to move to the target which is shown in Fig. 6b. Similar settings are also used in [17]. At the initial location, the robot takes 8 RGB images (45 degrees apart) using its head camera. Then it uses the existing semantic segmentation technique to obtain the semantic mask of each RGB image. After that, it predicts the top-down-view semantic map with our VPN. Finally, it parses the instruction and calculates the centroid coordinates of all "bench" pixels (cf. Fig. 6c).

Our real robot experiment shows that though the model is trained in a simulator it exhibits reasonable robustness when we randomly set the initial location and change the layout of surrounding objects. Example episodes are provided in the demo video.

VII. CONCLUSION

In this work, we propose the cross-view semantic segmentation task to sense the environment as well as a neural architecture design View Parsing Network (VPN) to address that. Based on the promising experimental results, we demonstrate that our VPN can be applied to mobilre robots to facilitate the surrounding awareness through a lightweight and efficient top-down-view semantic map. In many situations where the height information of objects is not that essential, VPN could be a good alternative compared to the traditional 3D-based methods which are costly on both data memory and computation.

REFERENCES

[1] I. Kostavelis and A. Gasteratos, "Semantic mapping for mobile robotics tasks: A survey," *Robotics and Autonomous Systems*, vol. 66, pp. 86–103, 2015.

[2] N. Sünderhauf, F. Dayoub, S. McMahon, B. Talbot, R. Schulz, P. Corke, G. Wyeth, B. Upcroft, and M. Milford, "Place categorization and semantic mapping on a mobile robot," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 5729–5736.

[3] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. CVPR*, 2016.

[4] Y. Katsumata, A. Taniguchi, Y. Hagiwara, and T. Taniguchi, "Semantic mapping based on spatial concepts for grounding words related to places in daily environments," *Frontiers in Robotics and AI*, vol. 6, p. 31, 2019.

[5] K. Zheng, A. Pronobis, and R. P. Rao, "Learning graph-structured sum-product networks for probabilistic semantic maps," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[6] A. Dai, A. X. Chang, M. Savva, M. Halber, T. A. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes." in *Proc. CVPR*, vol. 2, 2017, p. 10.

[7] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian, "Building generalizable agents with a realistic and rich 3d environment," *arXiv preprint arXiv:1801.02209*, 2018.

[8] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[9] 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.

[10] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. CVPR*, 2017.

[11] C. Zou, A. Colburn, Q. Shan, and D. Hoiem, "Layoutnet: Reconstructing the 3d room layout from a single rgb image," in *Proc. CVPR*, 2018.

[12] V. Hedau, D. Hoiem, and D. Forsyth, "Recovering free space of indoor scenes from a single image," in *Proc. CVPR*, 2012.

[13] S. Schulter, M. Zhai, N. Jacobs, and M. Chandraker, "Learning to look around objects for top-view representations of outdoor scenes," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 787–802.

[14] Z. Wang, B. Liu, S. Schulter, and M. Chandraker, "A parametric top-view representation of complex road scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 325–10 333.

[15] M. Zhai, Z. Bessinger, S. Workman, and N. Jacobs, "Predicting ground-level scene layout from aerial imagery," in *Proc. CVPR*, vol. 3, 2017.

[16] K. Regmi and A. Borji, "Cross-view image synthesis using conditional gans," in *Proc. CVPR*, 2018, pp. 3501–3510.

[17] W. B. Shen, D. Xu, Y. Zhu, L. J. Guibas, L. Fei-Fei, and S. Savarese, "Situational fusion of visual representation for visual navigation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2881–2890.

[18] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker, "Learning to adapt structured output space for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7472–7481.

[19] K. Sun, Y. Zhao, B. Jiang, T. Cheng, B. Xiao, D. Liu, Y. Mu, X. Wang, W. Liu, and J. Wang, "High-resolution representations for labeling pixels and regions," *arXiv preprint arXiv:1904.04514*, 2019.

[20] A. Murali, T. Chen, K. V. Alwala, D. Gandhi, L. Pinto, S. Gupta, and A. Gupta, "Pyrobot: An open-source robotics framework for research and benchmarking," *arXiv preprint arXiv:1906.08236*, 2019.