Predicting Dense and Context-aware Cost Maps for Semantic Robot Navigation

Yash Goel*[1,2], Narunas Vaskevicius*[2], Luigi Palmieri[2], Nived Chebrolu[3], Cyrill Stachniss[1,4]

***Abstract*— We investigate the task of object goal navigation in unknown environments where the target is specified by a semantic label (e.g. find a couch). Such a navigation task is especially challenging as it requires understanding of semantic context in diverse settings. Most of the prior work tackles this problem under the assumption of a discrete action policy whereas we present an approach with continuous control which brings it closer to real world applications. We propose a deep neural network architecture and loss function to predict dense cost maps that implicitly contain semantic context and guide the robot towards the semantic goal. We also present a novel way of fusing mid-level visual representations in our architecture to provide additional semantic cues for cost map prediction. The estimated cost maps are then used by a sampling-based model predictive controller (MPC) for generating continuous robot actions. The preliminary experiments suggest that the cost maps generated by our network are suitable for the MPC and can guide the agent to the semantic goal more efficiently than a baseline approach. The results also indicate the importance of mid-level representations for navigation by improving the success rate by 7 percentage points.**

I. Introduction

Equipping a robot with human-like semantic aware navigation skills is essential for achieving true autonomy. A robot tasked with finding a couch should be able to draw conclusion that if it is near a TV then the couch should be nearby - since they generally tend to be in the same space (*living room*). It is important to learn these semantic relationships between various objects in the environment and also between different spaces in the environment. This high level understanding can then be used to navigate the robot to target objects or area.

Reliable and accurate semantic robot navigation is still an open research question [6]. Traditional approaches use semantic knowledge for building graphs [12] or try to navigate to rooms [17] using various planning approaches, however those approaches tend to rely on hand-defined features and representations. The latter being built on top of various perception algorithms like object detection or semantic segmentation.

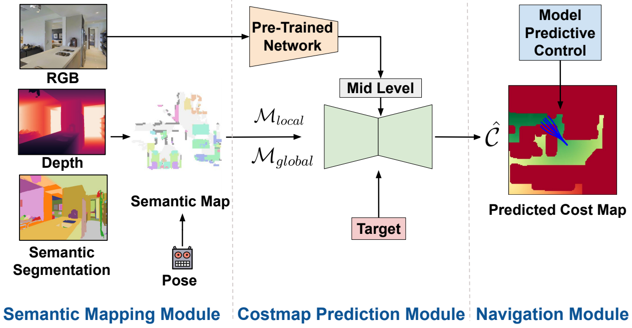

Fig. 1: Our framework is composed of three modules: *semantic mapping module* which constructs the map of the environment as the robot explores. The *cost map prediction module* which predicts the cost map for navigation based on the input semantic map, mid-level representation and target object. Finally, the *navigation module* generates the optimal control using a sampling-based MPC.

Recently with the surge of deep learning for computer vision and reinforcement learning various new methods have been proposed to tackle this problem. End-to-end control learning [5], [9], hybrid approaches combining traditional planning with RL [4], [8], among others have been proposed. Many of these works use reinforcement learning as a basis for their control and tend to use only a limited discrete action space. They focus on policies with simple actions like (left, right and straight) and do not address the problem of continuous robot motion control. With the goal of achieving efficient *object goal navigation*, i.e. reaching a defined semantic target, and fully exploit robot kinematics, inspired by [7], we propose a technique that combines model-based continuous control approach with perception module that exploits semantic information and mid-level feature representations. Differently from [11], instead of predicting only at the frontiers we provide dense predictions: the latter is a more natural and common representation for downstream tasks such as planning and control.

We summarize our main contributions as follows:

(i) We propose a U-Net based architecture for dense cost map prediction under partial observability and design loss function along with dataset for training.

(ii) We explore the use of egocentric mid-level visual representations in the network architecture. We present a novel approach of fusing these features to our network in an robot-orientation-aware way. Our experimental evaluation shows that mid-level representations significantly (by 7 percentage points in success rate) improve the navigation performance.

[1] Y. Goel is with the laboratory for Photogrammetry and Robotics, University of Bonn, Germany, and Robert Bosch GmbH. {s7yagoel}@uni-bonn.de.

[2] L. Palmieri, N. Vaskevicius are with Robert Bosch GmbH, Corporate Research, Stuttgart, Germany. {luigi.palmieri, narunas.vaskevicius}@de.bosch.com.

[3] N. Chebrolu is with with the Dynamic Robot Systems Group, University of Oxford, UK. {nived}@robots.ox.ac.

[1,4] C. Stachniss is with the University of Bonn, Germany, with the Department of Engineering Science at the University of Oxford, UK, and with the Lamarr Institute for Machine Learning and Artificial Intelligence, Germany. {cyrill.stachniss}@igg.uni-bonn.de.

*Denotes equal contribution.

This work was partly supported by the EU Horizon 2020 research and innovation program under grant agreement No. 101017274 (DARKO).

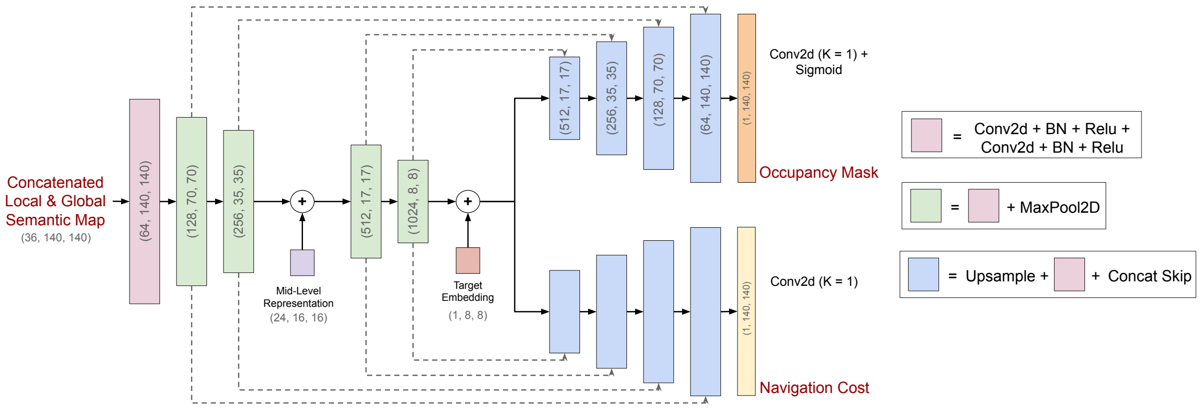

Fig. 2: Our detailed network architecture for cost map prediction. Dashed arrows denote skip connections. Kernel size K as 3, stride S as 1 and padding P as 1 are taken unless specified otherwise. The mid-level representations are fused according to orientation bin as shown in Fig. 3. In the end, the occupancy mask and navigation cost are fused to get the final cost map prediction.

(iii) We further show that the predicted cost maps can be used for semantic navigation in a loosely coupled scheme with sampling-based model predictive control (MPC) in continuous action space.

II. Our Approach

For object goal navigation task, the agent $\mathcal{A}$ is spawned in a random position in an uknown environment and is given a target object category $\mathcal{T}$ that the agent has to reach (e.g. a target category like *bed*). The agent has access to sensor observations ($\mathcal{O}_t$) consisting of ego-centric RGB, depth and semantic segmentation provided from the Habitat simulator [14]. Apart from this, the robot odometry, $\mathbf{x}_t$ is also available. The model of the robot is a differential drive model which is operated using velocity control (translational velocity v_t and angular velocity ω_t).

Our approach is shown in the Fig. 1. The whole framework can be divided into three major components. First component is *semantic mapping*, which is discussed in Sec. II-A. The output from this module is send to *cost map prediction network*, which is discussed in the Sec. II-B. It takes as input the unexplored and incomplete map around the robot to predict the cost map. The loss is defined in Sec. II-C. We use the predicted cost map in the *navigation* module as detailed in the Sec. II-D.

A. Semantic Map Generation

We follow the approach of Chaplot et al. [4] to construct the semantic map $\mathcal{M}$ of the environment. Overall, it contains the information of obstacles, explored area and top-down semantic information of each grid cell in the map. We start by transforming the first person semantic segmentation and depth to top-down semantic maps. We use the ego-centric semantic segmentation from the Habitat simulator [14] and keep the study of using off-the-shelf pre-trained segmentation network as a future work.

Using known camera intrinsic parameters and depth from Habitat, each pixel in the camera image is projected to 3D space along with their semantic label to get a semantic point cloud. The point cloud is then transformed to the world coordinates by using the robot pose $\mathbf{x}_t \in \mathcal{X}$, $\mathcal{X}$ being the space of all possible robot states. The semantic point cloud is converted to top-down 2D semantic map such that each cell has a semantic label with different probabilities for each class. For the K number of semantic classes we have the semantic map of size (K, N, N) where N is the size of local spatial region that we see in each view. We use 16 semantic classes which is a superset of our target classes to represent our semantic map. We further concatenate this map with obstacle mask and explored mask to finally get map $\mathcal{M}_t$ of size (C, N, N) where $C = 2 + K$ for time t. This unit generates a local and a global semantic map (see Sec. II-B).

B. Cost Map Prediction

We use the local semantic map, global semantic map and the mid-level representations [15] as the input to the network. The network architecture is inspired from the U-Net architecture [13]. The whole module architecture can be seen in the Fig. 2. We discuss the various components in this section.

Semantic Map. The local semantic map $\mathcal{M}_{local}$ and the global semantic map $\mathcal{M}_{global}$ generated from the *semantic mapping* module in Sec. II-A are used as input to the network. They are concatenated across the channel before being given as an input to the network. So, the total map input size is $(2C, H, W)$, which for our case is $(36, 140, 140)$. The global map is reduced to spatial size of local map using average pooling.

Mid-Level Representation. Mid-level representations are features generated from encoders which have been trained for different downstream tasks like *semantic segmentation*, *denoising*, *curvatures*, *keypoints*, etc. They have shown to be quite effective in RL setting for various downstream tasks [16], [15]: they improve generalizability and also performance of an RL agent. In our approach, we adopt mid-level representations and show how they can be beneficial for predicting context-aware dense cost maps.

The mid-level representations are extracted from the RGB image using a pre-trained network as used by Sax et al. [15].

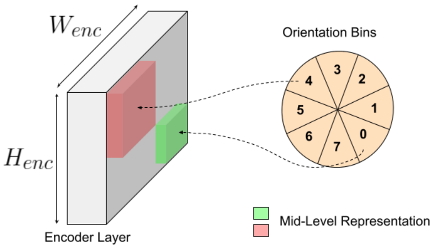

Fig. 3: Orientation of the robot decides the orientation bin in which the mid-level feature falls. Based on the bin, the corresponding region (bin region) is set to the mid-level feature while the rest of layer is set to 0 to form the oriented mid-level representation. Each bin is of the size 45° so that we have a total of 8 bins to cover all the orientation possibilities. For example, the agent looking towards the north will have the bin 3 and the associated middle top region will have the mid-level feature.

We use *semantic segmentation* and *object classification* to help learn about object goal and semantic contexts of the world. Apart from this, we use *depth prediction* since we are learning distance dependent cost map. These are combined at the encoder stage of the network. They are binned according to the orientation of the robot to form *oriented mid-level representation* as explained in Fig. 3. Using this binning technique we make these mid-level representations robot orientation aware and associate with the corresponding semantic input map region.

Target Embedding. We encode the target object category that the agent has to reach. To do this, we create an embedding using the index of the target category. The embedding of size M, where $M = 64$, is transformed to the spatial size of the encoded latent feature (H_{enc}, W_{enc}). It is then concatenated to the latent feature along the channel dimension.

Output. We observed that casting the cost map prediction as a multi-task learning problem leads to better results. Therefore, our network consists of two decoder branches. One predicts the occupancy map $\hat{\mathcal{C}}^{occ}$ and the other decoder branch predicts navigation cost $\hat{\mathcal{C}}^{nav}$. We combine $\hat{\mathcal{C}}^{occ}$ and $\hat{\mathcal{C}}^{nav}$ to get the final cost map prediction $\hat{\mathcal{C}}$. This is done by using an occupancy threshold θ_{occ} to create a binary occupancy mask from $\hat{\mathcal{C}}^{occ}$, which is then overlayed on the navigation cost $\hat{\mathcal{C}}^{nav}$.

C. Loss Function

In this section we describe our multi-term loss function that we formulated to train the network for the cost map prediction. In the following equations we denote the ground truth occupancy mask by $\mathcal{C}^{occ}$ and the ground truth navigation cost by $\mathcal{C}^{nav}$. The dataset generation technique for ground truth has been discussed in the Sec. III-A.

Occupancy Loss. We use a binary cross entropy loss over the local map region of size (H, W) to learn the occupancy probabilities $\hat{c}^{occ}_{i,j}$ of a map cell (i, j):

$$\mathcal{L}_{occ} = \frac{1}{HW}\sum_{i,j} -c^{occ}_{i,j}\log\left(\hat{c}^{occ}_{i,j}\right) - \left(1 - c^{occ}_{i,j}\right)\log\left(1 - \hat{c}^{occ}_{i,j}\right), \tag{1}$$

where $c^{occ}_{i,j}$ is 1 in case of obstacle and 0 in case of free space. This loss term is used only for the branch predicting the occupancy map.

Cost Map Loss. To learn the cost map prediction, we regress the navigation cost using the L1 norm, averaged over all valid positions (i.e. navigable area):

$$\mathcal{L}_{cost} = \frac{1}{HW}\sum_{i,j}\left\|\left(c^{nav}_{i,j} - \hat{c}^{nav}_{i,j}\right)\left(1 - c^{occ}_{i,j}\right)\right\|_1, \tag{2}$$

where $c^{nav}_{i,j}$ is the normalized ground truth navigation cost and $\hat{c}^{nav}_{i,j}$ is the predicted navigation cost.

Gradient Direction Loss. We introduced this term to make the navigation cost smooth and consistent with the local ground truth gradients. Similar to cost map loss, we only calculate this loss over navigable area of the map:

$$\mathcal{L}_{dir} = \frac{1}{HW}\sum_{i,j}\left(1 - \frac{\mathbf{g}_{i,j}\cdot\hat{\mathbf{g}}_{i,j}}{|\mathbf{g}_{i,j}|\cdot|\hat{\mathbf{g}}_{i,j}|}\right)\left(1 - c^{occ}_{i,j}\right), \tag{3}$$

where $\mathbf{g}$ is the gradient for ground truth cost map and $\hat{\mathbf{g}}$ is the gradient for predicted cost map,

$$\mathbf{g}_{i,j} = \left(\frac{\delta\mathcal{C}^{nav}}{\delta x}, \frac{\delta\mathcal{C}^{nav}}{\delta y}\right)_{i,j}$$
$$\hat{\mathbf{g}}_{i,j} = \left(\frac{\delta\hat{\mathcal{C}}^{nav}}{\delta x}, \frac{\delta\hat{\mathcal{C}}^{nav}}{\delta y}\right)_{i,j}.$$

In our experiments we observed that the addition of this term leads to significantly smoother cost maps, which is important for the downstream navigation task.

The total loss then becomes a combination of all these losses,

$$\mathcal{L}_{total} = \alpha_{occ}\mathcal{L}_{occ} + \alpha_{cost}\mathcal{L}_{cost} + \alpha_{dir}\mathcal{L}_{dir}, \tag{4}$$

with the empirically selected weights $\alpha_{occ} = 1.0$, $\alpha_{cost} = 1.5$ and $\alpha_{dir} = 1.0$.

D. MPC based Navigation

We use the sampling-based MPC approach of IT-MPC (Infomation Theoretic Model Predictive Control [18]) for the agent to pick the optimal control sequence. We start by having the initial control sequence $U = \{\mathbf{u}_0, \mathbf{u}_1, \dots \mathbf{u}_{H-1}\}$ where H is the horizon of the IT-MPC. Each timestep control, $\mathbf{u}_t = \{v_t, \omega_t\}$ where v_t and ω_t are the linear and angular velocity respectively for timestep t. Each control in the sequence is then perturbed for K samples to generate noisy control, $\tilde{U}_k = U + \mathcal{E}_k$ where $\mathcal{E}_k = \{\epsilon_0, \epsilon_1, \dots \epsilon_{H-1}\}$. Every noise ϵ_t is sampled from a normal distribution $\mathcal{N}(\mu, \sigma)$ using μ = 0 and σ = 0.35.

For each sample $\tilde{U}_k$, we generate the cost $\mathcal{S}_k$ using the predicted cost map and the control effort.

$$\mathcal{S}_k = \sum_{t=0}^{H-1}\left(\hat{\mathcal{C}}_t(\mathbf{x}_t) + \mathbf{u}_t^T Q\mathbf{u}_t\right), \tag{5}$$

where $\mathbf{x}_t = \{x_t, y_t, \theta_t\}$ is the robot pose and $Q \geq 0$ is the control effort matrix. The robot pose is sampled using a constant velocity differential model using the perturbed linear and angular velocity.

The cost is then used to generate weights w_k of the importance sampling step that obtains the optimal control sequence to execute: $\beta = \min_k S\left(\mathcal{E}^k\right), \eta = \sum_{k=0}^{K-1} \exp\left(-\frac{1}{\lambda}\left(S\left(\mathcal{E}^k\right) - \beta\right)\right), w_k = \frac{1}{\eta} \exp\left(-\frac{1}{\lambda}\left(S\left(\mathcal{E}^k\right) - \beta\right)\right)$. Finally, the control sequence $\mathbf{u}_t$ is updated using the weights and control noise.

$$\mathbf{u}_{t+1} = \mathbf{u}_t + \sum_{k=1}^{K} w_k \epsilon_t^k. \tag{6}$$

Hence, we get the updated control and we apply the first control $\mathbf{u}_0$ to the agent.

Goal Reacher. During the exploration, if the target object category $\mathcal{T}$ is observed in the local top-down semantic map $\mathcal{M}_{local,t}$ then the corresponding map cells define the goal mask $\mathcal{M}_{goal,t}$. To avoid false positives we remove small regions from the mask. Then using the remaining goal mask $\mathcal{M}_{goal,t}$ and local occupancy map $\mathcal{M}_{local,t}^{occ}$ we generate cost map for navigation using Fast Marching Method (FMM) [10]:

$$\mathcal{C}_t^{goal} = \text{FMM}\left(\mathcal{M}_{local,t}^{occ}, \mathcal{M}_{goal,t}\right). \tag{7}$$

The agent then drives using this cost map. It declares `done` when the cost map value is less than or equal to the cost threshold θ_{cost} i.e. $\mathcal{C}_t^{goal} \leq \theta_{cost}$. In our experiments we used $\theta_{cost} = 0.2$. If the goal turns out to be unreachable due to previously unobserved obstacles then our approach leaves the goal reaching mode and resorts to the predicted cost map to continue the exploration.

III. Experimental Setup

We perform experiments in the real-world indoor environments provided by a large-scale RGB-D dataset Matterport3D (MP3D) [3]. We use a physics-enabled 3D simulator Habitat [14] to navigate the agent in these environments. To train our cost map prediction network we generate a dataset as described in Sec. III-A. We describe the evaluation setup and the metrics for cost map prediction and navigation in Sec. III-B and Sec. III-C respectively. Finally, Sec. III-D provides important implementation details.

A. Cost Map Prediction Dataset

Our interest is in house-like environments containing objects, such as a couch, a bed, a table, a chair, a plant, etc. Therefore, as a first step, we filter out environments from MP3D dataset which do not contain relevant semantic information e.g. large halls or churches. In addition, we omit the houses containing multiple incorrect object labels, which can impair the training process. The remaining 48 houses form our dataset, which is divided into the train, validation and test splits consisting of 36, 4 and 8 houses respectively.

For each house, we sample multiple starting points and randomly select a goal from the set of goals we are considering. For each floor where the robot is spawned, we get the ground truth top-down semantic map as defined in [2] which is then used to generate the goal map based on the target object. Combining this goal map with the occupancy map from the Habitat simulator, we generate the global ground truth cost map of distances.

We use an IT-MPC based agent with the ground truth cost maps to reach the goal while we collect the dataset. We record samples at every fourth timestep to reduce redundancy. The number of trajectories taken in a house depend on the size of the house to avoid repetition. This was selected manually for each house upon inspection. The complete generated dataset contains 171412, 16209 and 41949 samples in the train, val, and test splits respectively.

Each training sample consist of i.) local semantic map ($\mathcal{M}_{local}$) of size 140×140, ii.) global semantic map ($\mathcal{M}_{global}$) of size 420×420 to help capture global context, iii.) ground truth cost map ($\mathcal{C}$) composed of distances to the goal using FMM [10] and occupancy map, and, iv.) egocentric RGB for computing the mid-level visual representations [15]. We also save the orientation of the robot along with the image.

B. Evaluation Setup for Cost Map Prediction

We evaluate both occupancy and cost map prediction for our approach. Both the predictions are evaluated on the test split of the generated dataset (Sec. III-A). The occupancy prediction uses classification metrics of mean F1 score (mF1) and mean Intersection over Union (mIOU) averaged over both free space and occupied space classes. We also report mean pixel accuracy (MPA) for occupancy prediction. For navigation cost prediction, we report average Action Prediction (aAP) which determines the accuracy of picking the right local policy normalised by navigable area. aAP$_5$ determines per-pixel accuracy of picking the correct action based on the lowest cost from 4 basic directions and being stationary. Similarly, aAP$_9$ measures the same but with 8 neighbours and the robot position. aAP$_9$ gives us a more accurate resolution as the agent can move in diagonal directions as well.

C. Evaluation Setup for Object Goal Navigation

For navigation performance we ran the agent on different houses from the test split in the Habitat simulator. There are a total of 8 test houses - for each house we sample 40 random starting positions for the robot along with a random target object to reach. We gather various metrics related to success of reaching the target object like success of reaching the goal, SPL [1], - which weights the success according to the path length determining its efficiency and DTS [4] i.e. Distance to Success, which measures how far is the agent from the success distance, which in our experiments was set to 1 m. We also measure the smoothness of the final robot trajectory using average acceleration and jerk. We run each experiment for 500 timesteps which is equivalent to 50 s. The target object can be selected from any of the target category list that we consider: *bed*, *chair*, *sink*, *plant* or *couch*.

D. Implementation Details

The cost map prediction network was implemented in PyTorch. During training, we applied data augmentation by randomly rotating (with a probability of 0.15) the input semantic maps and the target cost maps by 90°, 180° or 270°. We used the SGD optimizer and a constant weight decay factor of 0.01. In all experiments, the learning rate followed the cosine decay schedule with a warmup phase of 25 epochs with a peak and terminal learning rates being 15e-5 and 1e-5 respectively. All cost map prediction models were trained for 200 epochs. For the MPC we used an horizon $H = 50$.

IV. Experimental Results

A. Quantitative Results

In this section, we analyze the quantitative results of our approach for cost map prediction and object goal navigation.

Prediction. For occupancy mask prediction, we get an MPA of 78.76%, mF1 of 75.88% and mIOU of 62.93%. We observe that the F1 score of occupied region is 80.68% and that of free region is 71.08%. This shows that our occupancy mask prediction approach is inclined to predict the occupancy class better than the free space class. Similar trend was seen for IOU. For occupied region, it is 69.06% and for free region is 56.79%. We get scores for aAP_5 and aAP_9 as 37.45% and 33.39% respectively. An example of our prediction can be seen in Fig. 4.

Navigation. We compare our approach for object goal navigation with the following agents:

1. *Priviliged Random Agent*: It picks a random value from the allowable set of linear and angular velocity to use as an action, it is made *privileged* by providing the *goal reacher*, which is semantically informed. This helps us to see the improvement in exploration by our agent.

2. *MPC with GT cost map*: This agent has access to ground truth cost map which is then used for navigation by the IT-MPC. This agent declares the completion of episode when the goal is visible. This agent is useful to understand the upper bound on the performance.

As can be seen in Tab. I, our agent does not match the performance of the GT agent which is an expected outcome. However, we can clearly see that our agent outperforms the privileged random one. Our approach is able to reach the goal with more success of 0.447 (increase of 0.197) and higher SPL of 0.329 (increased by 0.121) which shows our approach is capable of more efficient exploration. This is further justified by a decrease in DTS of 2.342m - which shows it is able to reach nearer to the goal. Our agent also generates much smoother path as evident from average acceleration and average jerk.

B. Qualitative Results

In this section we present the qualitative results to show how our agent successfully completes the task of object goal navigation on one of the test sequence. Fig. 5 shows an example of a successful trajectory taken by our agent in reaching the target object. We can see how the agent progresses towards the goal as it builds the semantic map of the environment. The predicted cost map efficiently guides the agent near the goal area. At timestep 217 the goal reacher (described in Sec. II-D) gets activated as the target object (i.e. *plant*) becomes visible in the local semantic map. Finally, the cost map generated by the goal reacher drives the agent to the goal.

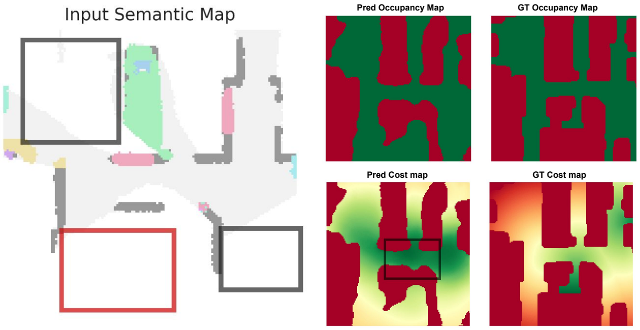

Fig. 4: The left image shows the input semantic map where, white is unexplored area, gray is free space and the rest is other semantic regions. The predicted occupancy map on the top right matches well with the ground-truth occupancy map in the black box regions. Whereas for the red box, our method is extrapolating free space. Here red is occupied and green is free space. For the cost maps on the bottom right, the predicted cost is able to capture the relative lower cost in the center of the map. For cost map, *red* to *green* represent *high* to *low* cost.

C. Ablation Study

In this ablation study, we compare how the different input information affect our results. We compare three variants: *Only Semantic Map*, which has only semantic map as the input to the cost map prediction module, *Only Mid-Level*, which has input map without semantics along with the mid-level representations and finally, *Our Approach*, with both semantic map and mid-level input. Results in Tab. II show that our approach achieves better results.

The *Only Mid-Level* already reaches a success rate of 0.402 while the one with *Only Semantic Map* has a success rate of 0.377. Addition of mid-level representations greatly improves the success rate of the *Only Semantic Map* by 7 percentage points. While addition of semantic map to *Only Mid-Level* achieves an improvement of 4.5 percentage points. This clearly shows that mid-level representations are beneficial for achieving better object goal navigation efficiency. A combination of both semantic representations leads to the best result in terms of success, SPL and DTS.

V. Conclusion

In this work, we present an approach to predict dense and context-aware cost maps for object goal navigation. Our proposed network architecture includes a novel way of fusing mid-level representations which takes into account the orientation of the robot. We demonstrate that the predicted cost maps can be used by sampling based MPC for semantic robot navigation. Moreover, the experiments indicate that the fusion of mid-level representations brings substantial improvement to the navigation performance. As future work, to better understand the potential of our approach, we will investigate additional semantic navigation baselines in continuous control settings. In addition, we wish to perform experiments on real robots and tackle uncertainties and noise as dense cost maps provide an appropriate base for that.

Methods	SR ↑	SPL ↑	DTS(m) ↓	Time ↓	Acc(ms^{-2}, $rads^{-2}$) ↓	Jerk(ms^{-3}, $rads^{-3}$) ↓
GT cost map	1.000	0.922	0.211	147.459	[0.15, 2.041]	[2.371, 35.006]
Privil. Random	0.250	0.208	7.214	402.795	[1.607, 8.920]	[28.004, 156.095]
Our Approach	**0.447**	**0.329**	**4.542**	**327.877**	**[0.673, 7.318]**	**[11.788, 128.158]**

TABLE I: Navigation performance comparison. GT cost map provides an indication of the best possible metrics.

Methods	*Cost Map Prediction*			*Object Goal Navigation*				
	MPA(%) ↑	mF1(%) ↑	mIOU(%) ↑	aAP$_5$(%) ↑	aAP$_9$(%) ↑	SR ↑	SPL ↑	DTS(m) ↓
Only Semantic Map	**79.17**	**76.32**	63.34	36.49	32.47	0.377	0.290	4.803
Only Mid-Level	79.11	**76.32**	**63.47**	37.21	33.11	0.402	0.287	4.716
Both (Our Approach)	78.76	75.89	62.93	**37.45**	**33.39**	**0.447**	**0.329**	**4.542**

TABLE II: Performance comparison for different ablations.

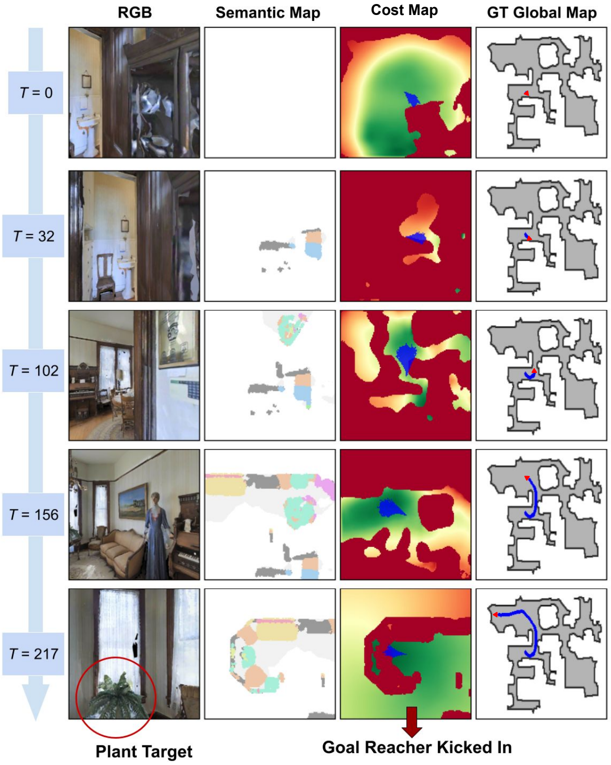

Fig. 5: Progression of agent moving in the house over time. The path over time is shown in blue in the GT global map with the red arrow showing the orientation of the robot. The target goal in this case is *plant* and we see that the agent is able to navigate to the plant efficiently. We also see in the last timesteps the goal reacher is activated. In the cost maps, the *green* regions show low cost and *red* the high-cost regions. Samples from MPC are also shown in the cost maps in *blue*.

References

[1] P. Anderson, A. Chang, D. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecká, J. Malik, R. Mottaghi, M. Savva, and A. Zamir. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

[2] V. Cartillier, Z. Ren, S. Jain, N.and Lee, I. Essa, and D. Batra. Semantic mapnet: Building allocentric semantic maps and representations from egocentric views. *arXiv preprint arXiv:2010.01191*, 2020.

[3] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niebner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *Proc. of the Int. Conf. on 3D Vision (3DV)*, 2018.

[4] D.S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, 2020.

[5] T. Chen, S. Gupta, and A. Gupta. Learning exploration policies for navigation. In *Proc. of the Int. Conf. on Learning Representations(ICLR)*, 2019.

[6] J. Crespo, J.C. Castillo, O.M. Mozos, and R. Barber. Semantic information for robot navigation: A survey. *Applied Sciences*, 10, 2020.

[7] P. Drews, G. Williams, B. Goldfain, E. Theodorou, and J. Rehg. Aggressive deep driving: Combining convolutional neural networks and model predictive control. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2017.

[8] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive Mapping and Planning for Visual Navigation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[9] A. Mousavian, A. Toshev, M. Fiser, J. Kosecká, A. Wahid, and J. Davidson. Visual representations for semantic target driven navigation. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 8846–8852, 2019.

[10] S. Osher and J.A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.

[11] S.K. Ramakrishnan, D.S. Chaplot, Z. Al-Halah, J. Malik, and K. Grauman. Poni: Potential functions for objectgoal navigation with interaction-free learning. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[12] J.G. Rogers and H.I. Christensen. Robot planning with a semantic map. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013.

[13] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[14] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A platform for embodied ai research. In *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, pages 9338–9346, 2019.

[15] A. Sax, B. Emi, A.R. Zamir, L.J. Guibas, S. Savarese, and J. Malik. Mid-level visual representations improve generalization and sample efficiency for learning visuomotor policies. *arXiv preprint arXiv:1812.11971*, 2018.

[16] B. Shen, D. Xu, Y. Zhu, L.J. Guibas, F. Li, and S. Savarese. Situational fusion of visual representation for visual navigation. In *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, pages 2881–2890, 2019.

[17] B. Talbot, O. Lam, R. Schulz, F. Dayoub, B. Upcroft, and G. Wyeth. Find my office: Navigating real space from semantic descriptions. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2016.

[18] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. Rehg, B. Boots, and E. Theodorou. Information Theoretic MPC for Model-Based Reinforcement Learning. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2017.