Spatial Relation Graph and Graph Convolutional Network for Object Goal Navigation

D. A. Sasi Kiran*[1], Kritika Anand*[2], Chaitanya Kharyal*[1], Gulshan Kumar[1]
Nandiraju Gireesh[1], Snehasis Banerjee[2], Ruddra dev Roychoudhury[2], Mohan Sridharan[3]
Brojeshwar Bhowmick[2], Madhava Krishna[1]

[1]Robotics Research Center, IIIT Hyderabad, India
[2]TCS Research, Tata Consultancy Services, India
[3]Intelligent Robotics Lab, University of Birmingham, UK

***Abstract*— This paper describes a framework for the object-goal navigation task, which requires a robot to find and move to the closest instance of a target object class from a random starting position. The framework uses a history of robot trajectories to learn a Spatial Relational Graph (SRG) and Graph Convolutional Network (GCN)-based embeddings for the likelihood of proximity of different semantically-labeled regions and the occurrence of different object classes in these regions. To locate a target object instance during evaluation, the robot uses Bayesian inference and the SRG to estimate the visible regions, and uses the learned GCN embeddings to rank visible regions and select the region to explore next. This approach is tested using the Matterport3D benchmark dataset of indoor scenes in AI Habitat, a visually realistic simulation environment, to report substantial performance improvement in comparison with state of the art baselines.**

***Index Terms*— Spatial Relational Graph, Graph Convolutional Networks, Semantic Object Navigation.**

I. Introduction

Navigation is a fundamental task performed by a service robot, e.g., in an office or a home. Navigation tasks are broadly classified into *PointGoal* tasks (go to a point in space), *ObjectGoal* tasks (go to a semantically distinct object instance), and *AreaGoal* tasks (go to a semantically distinct area) [1]. This work focuses on *ObjectGoal* navigation tasks, also called *ObjectNav*. As a motivating example, consider a service robot equipped with a camera, which has been asked by a human to go to a 'sink' in a home. It is difficult for the robot to perform such a task that humans perform effortlessly. The robot needs to process sensor inputs, understand its environment, and make suitable decisions to move to the target. Specifically, to go to a 'sink', the robot needs to know that a 'sink' is an object usually found in a region labeled 'kitchen'. It also needs to confirm its current location based on observations of *relevant* objects in view, e.g., it is in the 'bedroom' because it sees a 'bed' nearby.In addition, it needs to use knowledge of the environment and the objects in its view to estimate regions that are traversable, and select the region that is most likely to lead to the target object. In the current example, the robot knows it is very unlikely to find a 'sink' in the 'bedroom', and decides to move to the 'living room', an adjacent region. The robot does not find a sink in the living room but it does observe an 'oven' in a nearby region. It reasons that the region is most probably a 'kitchen' because that is where an oven is most likely to be found. Since the robot knows that a kitchen is very likely to contain a sink, it decides to move to the corresponding region where it finds a sink. Our framework makes the following novel contributions towards realizing this motivating scenario:

- An approach that uses the robot's trajectories in similar environments to learn a *Spatial Relational Graph* (SRG) that models the probability of proximity of different semantically-labeled regions to each other and the occurrence of specific object classes in each region.
- An approach that uses a *Graph Convolutional Network* (GCN) operating on the historical trajectories and the learned SRG to learn the *embeddings* of each region and object based on their co-occurrence.
- A Bayesian inference approach that uses the SRG during evaluation to incrementally process the robot's current observations of specific objects and to estimate the labels of the regions visible in its current location.
- An approach that uses the GCN-based embeddings to select the visible region to explore next, computing for each region the likelihood of leading to a region with an instance of the target object class.

We use off-the-shelf algorithms for planning a path and moving a robot to a desired location and abstract away the object recognition task by assuming accurate recognition of observed objects in images of any given scene. The framework is evaluated using benchmark indoor scenes from the Matterport3D (MP3D) dataset [2] and baseline methods in the visually realistic AI Habitat simulation environment [3]. A marked improvement in relevant measures in comparison with state of the art baselines is shown. Additional results and supporting material are available online: `https://user432.github.io/objnav-srg/`.

II. Related Work

We review related work on the ObjectNav task, focusing on state of the art data-driven methods.

*Denotes equal contribution

Mapping based approaches: The use of data-driven methods to learn a semantic map or an occupancy map to assist in the ObjectNav task continues to be a popular approach. These methods often use a dedicated module or a (deep) neural network, e.g., the use of a neural network to obtain a mao that is then used to sample a long-term goal to guide exploration [4], [5], [6]. There has also been work on using a neural network to estimate occupancy for the related PointNav task, i.e., to reach a point instead of an instance of an object class [7]. Instead of relying on a map, our framework uses a spatial relational graph and embeddings of the visible regions and objects to guide exploration.

End-to-End approaches: Data-driven methods have been developed to directly move to a given goal based on sensor inputs by learning to predict actions instead of building multiple linked components [8], [9]. This includes the use of Reinforcement Learning (RL) methods [10], [11].

Graph based approaches: Relational graphical models have been trained and used to select actions for navigation [12], [13], [14], [15]. One method builds a relational graph during training to encapsulate the relational dependencies between different regions in the scene [12]. This graph is updated periodically during testing using a Convolutional Neural Network(CNN)-based region predictor network. Another method builds a topological graph during exploration, with nodes representing the locations that are used to select sub-goals [13]. There has also been work on building a graph with region nodes, zone nodes, and object nodes, with one of the zone nodes being selected as the sub-goal that is reached using RL methods [14]. In another method, the graph's nodes are a few landmarks objects and robot poses, and an RL agent is trained to navigate to all possible objects [15]. There are also methods that exploit graphical relations in different ways to aid navigation [16], [17], [18]. Many of the methods discussed above focus on specific simulators or datasets and make corresponding assumptions. Our framework uses a relational graph as well, but to capture the relational dependencies between both the regions and the objects during training, and make decisions during evaluation. It is also used to do region prediction based on a probabilistic model which exploits this graph.

III. Problem Specification and Methodology

We focus on the *ObjectNav* task in which a robot placed in a random pose in a previously unknown indoor environment is asked to find an instance of a target object class [19]. Figure 1 is an overview of our framework that has six stages. The first three stages, described in Sections III-A- III-C, correspond to the training process during which the robot executes and uses trajectories of its movement through a set of semantically-labeled scenes to compute the SRG and the GCN embeddings. The next three stages, described in Sections III-D- III-F, correspond to evaluation during which the robot uses the trained SRG and GCN to process input observations and compute a ranking of the visible regions in terms of their likelihood to lead to an instance of the target object class; an off-the-shelf planner is then used to control the robot's movement to the highest-ranked visible region. Individual stages are described below.

A. Generating Valid Trajectories

Figure 2 is an overview of the first step of the training process. The robot is initialized in a known *MP3D* scene and given a goal to find the nearest instance of an object class. The environment is known to the robot and it moves to all the instances of the object category keeping a record of the regions encountered along the path. The path with the minimum distance to the target object is labeled as a *valid trajectory* and stored. We generate multiple valid trajectory paths for subsequent use.

As an example, consider a trial in which the target object is a 'sink' with the robot initialized in the region 'bedroom'. There are instances of a sink in multiple regions of the *MP3D* house/scene such as the *bathroom*, *laundry room*, and *kitchen*; the nearest *sink* to the *bedroom* is in the *bathroom*. If the robot instead starts in the 'living room' with the same target object class (i.e., 'sink'), the valid trajectory may be $\{$*living room*, *hallway*, *dining room*, *kitchen*$\} \rightarrow$ *sink*, i.e., the nearest 'sink' to the 'living room' is in the 'kitchen'. To get to the goal target object, we find the next action using *shortest path follower* algorithm, which takes into consideration the geodesic shortest path from the agent's current position to the goal position. Overall, we obtained $18,488$ trajectory paths in AI Habitat MP3D environment.

B. Generation of Spatial Relational Graph

The proposed SRG graphically represents the information about spatial relations between regions and objects, which is essential for object goal navigation task. We denote this graph by $G = (V, E)$, where V and E represent the nodes and the edges between nodes, respectively. In particular:

- Each node $n \in V$ denotes an *object* category (*object node*) or the *region* category (*region node*); and
- Each edge $e \in E$ denotes the relationship between *region* categories or between a *region* and an *object* category.

We consider 2 types of edges to encode: (i) '*Includes*' relation between a *region* and an *object* category; and (ii) '*Adjacency/proximity*' relation between a pair of regions.

In any MP3D scene with n regions, the robot is allowed to move from region R_i to other regions R_j, i.e., $i, j \in [1, \dots n]$, $i \neq j$. For each scene, an edge is created between region nodes n_{R_k} and n_{R_l} if nodes representing R_k and R_l are adjacent in the path between R_i and R_j. Also, we create an edge between object node n_o and a region node n_{R_i}, if object o is in region R_i. We create such *scene graphs* $(G_1, G_2, G_3, \dots, G_m \in \mathbf{G})$ for all m scenes in the MP3D dataset. These graphs in $\mathbf{G}$ are used to build the spatial relational graph (G_s) as depicted in Figure 4.

G_s encodes the proximity and spatial co-occurrence (frequency) statistics of *regions* and *objects* extracted from the MP3D scenes explored in the valid trajectories. Specifically, the SRG associates an attribute '*weight*' with the two different types of edge between the nodes; it represents the object-region co-occurrence probability for the *includes* relation

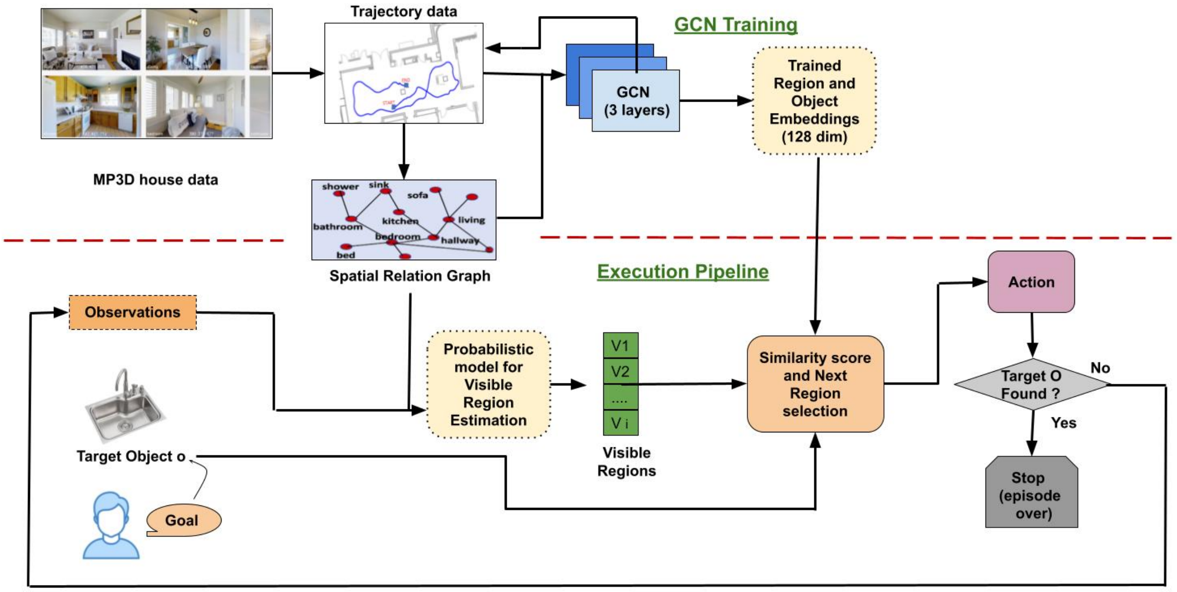

Fig. 1: Our ObjectNav framework has three stages each for training and evaluation. It trains and uses an SRG encoding the proximity of regions to each other and object-region co-occurrence, and a GCN-based embedding of this information and historical data of executed trajectories in indoor environments, to identify and move to the region most likely to contain an instance of the target object class in a previously unseen environment.

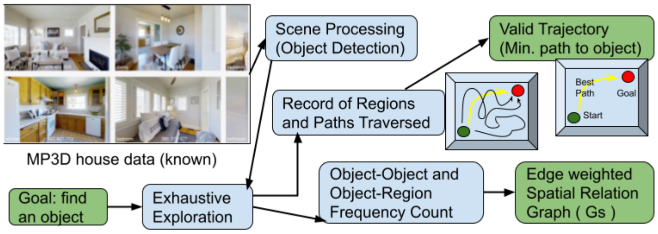

Fig. 2: Generating valid trajectories by exploring target object classes in MP3D dataset scenes using AI Habitat.

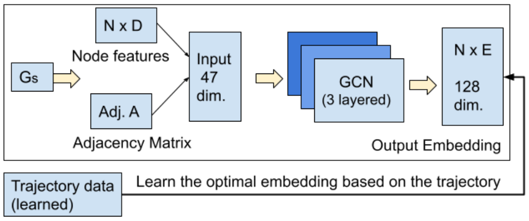

Fig. 3: Training GCN to encode embedding of information in SRG based on 'valid' trajectories.

between a particular object and a particular region, and the proximity likelihood for the *adjacency/proximity* relation between any two regions. These weights are computed for the individual graphs corresponding to the MP3D scenes $(G_1, G_2, G_3, \ldots, G_m)$. For example, to estimate the weight of an edge connecting object node $(n_{o=bed})$ and region node $(n_{R_i=bedroom})$ in G_s, we find the frequency of '*bed* in *bedroom*' in the graphs in $\mathbf{G}$ and normalize it by the frequency of '*bedroom*'. The *weight* for the edge between any two regions r_i and r_j is computed by dividing the total number of co-occurrences of r_i adjacent to r_j by the minimum of the frequency of the individual regions in $\mathbf{G}$.

Figure 4 shows an example SRG constructed from the scene graphs. We observe that the *weight* is high (0.89) for '*bed* in *bedroom*' and for '*bedroom* adjacent to *bathroom*' (*weight* = 0.87), whereas it is low (0.05) for '*bed* in *kitchen*' and '*bathroom* adjacent to *kitchen*' (*weight* = 0.35). These priors will help the robot discover and navigate towards the *target* object from its current position. For example, seeing a *dining table* from the *living room* will help the robot identify and navigate to an adjacent *kitchen* that is likely to contain a *sink*, which is the robot's target.

C. *Encoding Object-region Embeddings in GCN*

The previous section described how the SRG probabilistically encodes the proximity of regions to each other and the co-occurrence of objects and regions. For the 'ObjectNav' task, it is useful to obtain a low-dimensional embedding of this information and the useful trajectories contained in the valid trajectories collected during training.

Specifically, the objective is to train embeddings such that regions and objects more likely to occur together on the path to the target object have a high similarity score based on the embeddings. These embeddings are learned from the SRG and the positive trajectories by optimizing a cross-entropy loss function. Since positive trajectories represent the path taken by the robot in a known indoor environment to successfully locate the target object, the learned embeddings are similar to the Word2vec representation for computing an embedding for words in a sentence [20] .

We use a GCN to learn the embeddings because it is well-suited to capture the relationships in the SRG and trajectories. As before, we assume that the robot is able to correctly recognize objects in any observation of the current scene. Recall that the trained SRG G_s has two types of edges. As we are only interested in the most likely links in G_s, edges that have a *weight* ≤ 0.5 are pruned.

The GCN takes two inputs during training: (i) input features for every node i, represented as a $N \times D$ matrix (N: number of nodes, D: number of input features); and (ii) graph structure in the form of an adjacency matrix A of size $N \times N$ [21]. It produces an output of dimension $N \times E$ where E is the dimension of the embedding. The *region* and *object* categorical values are mapped to integer values using the *one-hot encoding vector* to avoid bias, i.e., the index of the node has value 1 and other values are zeros. Specifically, a three-layer GCN takes as input the SRG in the form of an *adjacency matrix* and an one-hot encoding of the features of region and object nodes. The dimension of feature vectors is the sum of the number of *objects* and *regions*; in this paper, we consider 19 objects and 28 *regions*. For training, we use the graph convolutional operator (GCNConv) [21]; the first layer has input dimension 47 and the last layer's output dimension is the embedding size (128 in this paper).

For every index $x \in \{2, 3, \cdots, n-1\}$ in a valid trajectory $\{i_1, i_2, i_3, \cdots i_n\} \rightarrow i_{\text{target}}$, we find its prefix path $\{i_1, i_2, i_3, \cdots, i_{x-1}\}$. In the loss function, we maximize the similarity of the embedding of node (i.e., region) i_x with the embedding of i_n, and with the embedding of each node in its prefix path. Suppose the robot took the path {*living room*, *hallway*, *bedroom*} → *bed* to reach the nearest instance of object class *bed* from its starting region. We maximize the similarity of the embedding of node *hallway* with node *bedroom*, and *hallway* with *living room*. Also, for each trajectory, we maximize the embedding of i_n with i_{target}, i.e., *bedroom* with target *bed* in the current example.

For each valid prefix path, we also generate invalid prefix paths in which the intermediate nodes are $i_{invalid}$ = R - i_{valid}, where R is the set of regions in the dataset and i_{valid} is the set of nodes (i.e., regions) in the trajectory to the target object. For example, if the *prefix path* p_1 is: {*living room*, *hallway*, *bedroom*}, the invalid prefix path will be {*living room*, *x*, *bedroom*}, where $x \in i_{invalid}$, i.e., {*living room*, *bathroom*, *bedroom*}, {*living room*, *dining room*, *bedroom*}, $\cdots$, {*living room*, *stairs*, *bedroom*}. For an invalid prefix path $p_{invalid} = \{i_1, i_2, i_3, \cdots, i_x\}$ and every index j $\in \{2, 3, ..x - 1\}$, we minimize the similarity of the embedding of node i_j with the embedding of i_{target}. During evaluation in a new scene, the embedding helps select a path most similar to the valid trajectories in the trained model.

D. Visible Region Estimation using SRG

Algorithm 1 Computing visible regions using SRG.

```
Input: SRG, $O_v = \{o_1, o_2, \cdots, o_l\}$
Output: $V$
visible_regions = []
for obj in $O_v$ do
    cand_objs = obj $\cup$ nearest_objects(obj, $O_v$, $k$=4 )
    compute probabilities $p(R_i|\text{cand\_objs})\ \forall i \in [1, n]$
    visible_regions[obj]= $\text{argmax}_i\{p(R_i|\text{cand\_objs})\}$
end for
return  visible_regions
```

During evaluation (i.e., testing), the robot has to use the learned SRG and GCN embeddings to reach an instance of a target object class in a previously unseen scene. To do so, the robot first identifies the visible regions based on the observed objects and the SRG. We use Bayesian inference and some simplifying assumptions to compute the probability of a region R being visible given a set of visible objects $O_v = \{o_1, o_2, \cdots o_n\}$. Without loss of generality, assume that the robot has observed two objects o_j and o_k. Then:

$$p(R_i|o_j, o_k) = \frac{p(R_i, o_j, o_k)}{p(o_j, o_k)} = \frac{p(o_j|R_i, o_k) \cdot p(R_i, o_k)}{p(o_j, o_k)} \tag{1}$$

where $p(R_i|o_j, o_k)$ is the probability of region R_i being in the robot's view given the objects $o_j, o_k \in O_v$. If we make the simplifying assumption that the presence of each object is independent of the other objects, we obtain:

$$\begin{aligned} p(o_j|R_i, o_k) &= \frac{p(o_j, R_i, o_k)}{p(R_i, o_k)} = \frac{p(o_j, o_k|R_i) \cdot p(R_i)}{p(R_i, o_k)} \\ &= \frac{p(o_j|R_i) \cdot p(o_k|R_i) \cdot p(R_i)}{p(o_k|R_i) \cdot p(R_i)} \\ &= p(o_j|R_i) \end{aligned} \tag{2}$$

which leads us to:

$$p(R_i|o_j, o_k) = \frac{p(o_j|R_i) \cdot p(o_k|R_i) \cdot p(R_i)}{p(o_j, o_k)} \tag{3}$$

Since $p(o_j, o_k)$ will be a factor in the probability computation of any region, it can be treated as a constant scaling factor. We also make an assumption that all the regions are equally likely initially, leading to:

$$p(R_i|o_j, o_k) = \lambda \cdot (p(o_j|R_i) \cdot p(o_k|R_i)) \tag{4}$$

where λ is a constant, and we get $p(o_j|R_i) \cdot p(o_k|R_i)$ from the 'includes' edges of the SRG. This computation can be performed incrementally to consider any number of observed objects in the scene, to obtain the region probabilities:

$$p(R_i|o_j, \cdots, o_k) \quad \forall i \in [1, n] \tag{5}$$

While estimating the visible region list, for every visible object, we consider the set of candidate objects $\{o_j, \cdots, o_k\}$ to also include k (experimentally set as 4) of its closest visible objects. This set is used to compute the region probabilities in Equation 5. The region label assigned to the visible object is that of the region with the highest probability in the vector above. Algorithm 1 summarizes these steps.

E. Identifying Next Region to Explore

Among the visible regions, the robot needs to select the region to explore next. Suppose that the robot is currently able to view regions V : $\{v_{r_1}, v_{r_2}, v_{r_3}, ..., v_{r_l}\}$. The robot uses

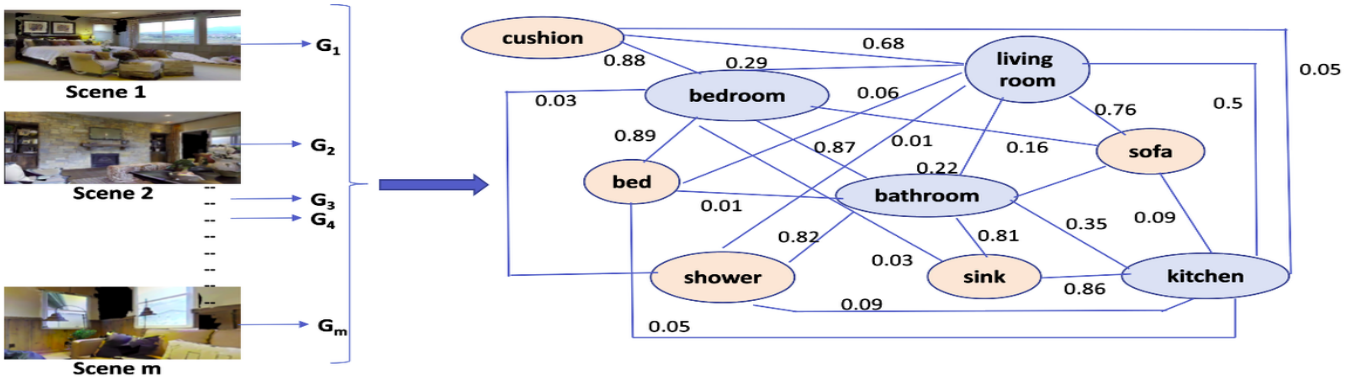

Fig. 4: Learning SRG from valid trajectories obtained through exploration of MP3D scenes. Nodes represent regions (blue) or object classes (orange), and edges encode likelihood of proximity (between regions) or occurrence of objects in regions.

the trained GCN to compute the embedding of each visible region and the similarity of these embeddings with those of the *target* object *t*. The region with the highest similarity is chosen to be explored next.

$$\text{Choose } \underset{v_{r_i}}{\operatorname{argmax}} \Big(Sim(Emb(t), Emb(v_{r_i})) \Big)|_{i\in[1,l]} \qquad (6)$$

where *Sim*() is the cosine similarity function and *Emb*() is the embedding output from the GCN.

F. Action Controller

After reaching the new region, if the agent sees the target object, it moves towards the object using a shortest path follower. If the target is within $1m$ Euclidean distance to the agent, the episode is terminated as a success. If the target object is not present in the new region a new set of visible regions is computed and the process is repeated until the *target* is found or the maximum number of steps (350 in this work) for the episode is reached. One step corresponds to a translation movement of 0.3 m forward or backward, or a 30^0 rotation to the left or right. An existing off-the-shelf planner is used for planning and executing local navigation.

IV. Experimental Setup

As baselines for comparison, we used three strategies: (i) Random action selection ; (ii) Active Neural Slam (ANS) [6]; and (iii) Graph convolutional region estimator network (GC-Exp) [22]. In ANS, the long term goal of the robot was chosen such that the exploration policy tried to maximize the area explored. A key component of GCExp was a region classification network (RCN), a graph neural network that mapped a Semantic graph $\mathcal{S}_t$ with objects from any of the N_D object classes as nodes, to a probability distribution over the N_R regions for each node. Its inputs include:

- Feature vector matrix $X \in \mathbb{R}^{N\times N_D}$ for node representation, where N is the number of nodes in $\mathcal{S}_t$. For each node n, the input feature vector $x_n \in \mathbb{R}^{N_D}$ is a one hot encoding of its object category.
- Adjacency matrix $A \in \mathbb{R}^{N\times N}$ of the graph structure.

We experimentally evaluated the following hypotheses:

H1: The proposed framework substantially improves the success rate compared with the above baselines.

H2: The use of SRG for visible region estimation provides performance comparable with the use of the RCN in our framework, while significantly reducing the computational effort.

H3: The SRG-based approach improves transparency in visible region estimation by explicitly identifying the objects influencing this estimation.

Hypotheses **H1** and **H2** were evaluated quantitatively while **H3** was evaluated qualitatively. Evaluation of hypotheses **H1-H2** was based on four well-established measures taken from related literature [1], [22], [23] :

1) **Success**: ratio of the number of successful episodes to total number of episodes. An episode is successful if the robot is ≤ 1.0 m from the target object.
2) **SPL** (Success weighted by path length): measures the efficiency of path taken by robot compared with optimal path; it is is computed as:

$$SPL = \frac{1}{N}\sum_{i=1}^{N} S_i . \frac{l_i}{max(p_i, l_i)}$$

where N is the number of test episodes, S_i is a binary success indicator, l_i is the length of shortest path to the closest instance of target object from the robot's initial position, and p_i is the length of path traversed by robot.
3) **SoftSPL**: it replaces the binary S_i from SPL with a continuous success indicator $\in [0, 1]$ depending on robot's distance to the goal.

$$SoftSPL = \frac{1}{N}\sum_{i=1}^{N} \underbrace{(1 - \frac{d_i}{max(l_i, d_i)})}_{episode\_progress} . (\frac{l_i}{max(p_i, l_i)})$$

where N, l_i, and p_i are as before, and d_i is the length of the shortest path to the goal from the robot's position at episode termination.
4) **Distance to Success (DTS)**: denotes the distance between the agent and the permissible distance to target for success at the end of an episode.

$$DTS = max(\|x_T - G\|_2 - d, 0)$$

where $\|x_T - G\|_2$ is the $L2$ distance between robot and goal at the end of the episode; d is the success threshold.

TABLE I: Comparing 'Success' of proposed framework with baselines. Proposed framework provides better performance than baselines; use of SRG instead of RCN for region estimation provides comparable performance at much lower computational effort.

Method	**Success↑**
Random	0.0056
ANS	0.69
ANS+GCExp	0.72
Framework with RCN	**0.773**
Framework with SRG	**0.751**

TABLE II: Comparing proposed framework and baselines on all four measures, focusing on the comparison between SRG and RCN for region estimation; use of SRG provides comparable performance at much lower computational effort.

Method	**Success ↑**	**SPL ↑**	**SoftSPL ↑**	**DTS (m) ↓**
Random	0.0056	0.0032	0.0751	7.1565
Framework+RCN	**0.773**	**0.548**	**0.565**	**1.993**
Framework+SRG	**0.751**	**0.530**	**0.553**	**2.348**

As stated earlier, training and evaluation used different sets of scenes from the Matterport3D (MP3D) benchmark dataset for ObjectNav task, within the visually realistic AI Habitat simulation environment. The proposed framework is trained on the trajectories taken from 51 scenes and the testing is done on 250 episodes (each) on five MP3D scenes.

V. EXPERIMENTAL RESULTS

To evaluate hypotheses **H1**-**H2**, we first compared our proposed approach with the baselines in terms of the 'Success' measure, with the corresponding results summarized in Table I. We observe that the proposed approach performed better than the baselines, e.g., top three rows of table compared with last row. The use of SRG for visual region estimation provided performance comparable to the use of RCN for visual region estimation in our framework; there was no significant qualitative difference in the results, but incremental Bayesian inference with SRG involved much less computational effort than training and testing the substantially more complex deep network structure of RCN.

Next, we compared the proposed framework with the baselines using all four measures, focusing on the comparison between SRG and RCN for region estimation. The results summarized in Table II indicate that SRG and RCN provide comparable performance. However, RCN involves computationally expensive training and use of a deep network for visible region estimation. SRG, on the other hand, supports incremental and efficient region estimation. Tables III-IV summarize results of a similar comparison on some representative scenes from the MP3D benchmark dataset.

We also explored the sensitivity of the framework's performance to the values of key parameters. Recall that we fix the radius d of the space within which we consider visible objects as $10m$. We also set the maximum number k of nearby objects that we considered for each object as 5 (Section III-D). The framework's performance for other specific values of these parameters is summarized in Tables V-VI.

TABLE III: Our framework's performance on **specific scenes** in MP3D with **SRG for region estimation**.

Scene	**Success ↑**	**SPL ↑**	**SoftSPL ↑**	**DTS (m) ↓**
17DRP5sb8fy	0.864	0.609	0.622	0.52
rPc6DW4iMge	0.700	0.494	0.520	2.61
S9hNv5qa7GM	0.616	0.393	0.403	2.67
b8cTxDM8gDG	0.868	0.675	0.689	1.25
EDJbREhghzL	0.708	0.491	0.529	4.69
Average	0.7512	0.530	0.553	2.348

TABLE IV: Our framework's performance on **specific scenes** in the MP3D with **RCN for region estimation**.

Scene	**Success ↑**	**SPL ↑**	**SoftSPL ↑**	**DTS** (m) ↓
17DRP5sb8fy	0.878	0.626	0.639	0.72
rPc6DW4iMge	0.714	0.496	0.516	2.52
S9hNv5qa7GM	0.532	0.335	0.345	3.56
b8cTxDM8gDG	0.936	0.690	0.697	0.725
EDJbREhghzL	0.806	0.595	0.631	2.443
Average	0.7732	0.548	0.565	1.994

Finally, to evaluate **H3**, we qualitatively compared the use of SRG-based region estimation with the use of RCN within our framework. Our framework improves transparency by providing a readily interpretable list of objects influencing the decision about specific visible regions; we also obtain a probability distribution over the candidate regions. Figure 5 shows an example of region estimation on a particular image and Table VII shows additional examples. An added advantage of using the SRG for region estimation is the reuse of the model in very different scenes if the distribution of objects over the semantic regions is similar.

Additional results and supporting material are available online: `https://user432.github.io/objnav-srg/`. Since our framework is designed for the ObjectNav task, it may not provide high accuracy as a generalized navigation module due to variations in the environment and the objective functions. However, experimental results indicate that extensive training on representative and realistic scenes leads to good performance on previously unseen scenes from similar environments.

TABLE V: Performance of our framework with SRG based Probabilistic Estimation Model; for d=7m.

Scene	**Success ↑**	**SPL ↑**	**SoftSPL ↑**	**DTS (m) ↓**
17DRP5sb8fy	0.82	0.58	0.60	1.01
rPc6DW4iMge	0.65	0.43	0.45	3.17
S9hNv5qa7GM	0.52	0.338	0.353	3.58
b8cTxDM8gDG	0.77	0.57	0.59	2.03
EDJbREhghzL	0.54	0.38	0.43	6.9
Average	0.66	0.4596	0.4846	3.338

TABLE VI: Performance of our framework with SRG based Probabilistic Estimation Model; for k=7.

Scene	**Success ↑**	**SPL ↑**	**SoftSPL ↑**	**DTS (m) ↓**
17DRP5sb8fy	0.82	0.60	0.61	0.87
rPc6DW4iMge	0.632	0.427	0.45	2.7
S9hNv5qa7GM	0.55	0.388	0.4	3.39
b8cTxDM8gDG	0.768	0.56	0.58	1.96
EDJbREhghzL	0.7	0.50	0.53	4.8
Average	0.694	0.495	0.514	2.744

TABLE VII: **Visible region estimations:** Outputs of SRG based visible region estimation model, over varied objects.

Objects in FOV	**Estimated Region**
‘cushion’,‘bed’,‘chair’ ‘cabinet’,‘cushion’	Bedroom
‘towel’,‘shower’,‘sink’ ‘towel’,‘chair’	Bathroom
’sofa’, ’sofa’, ’cushion’ ’table’, ’picture’	Living Room
‘table’,‘chair’,‘chair’ ‘chair’,‘picture’	Meeting room / Conference room
‘counter’,‘cabinet’,‘sink’	Bar
‘gym_equipment’,‘towel’,‘stool’ ‘gym_equipment’,‘cabinet’	Gym / Exercise room

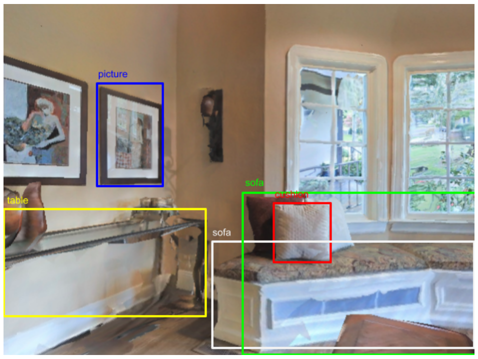

Fig. 5: **Visible region estimation: Living Room**. The objects seen by the model leading to the region estimation of ‘Living Room’ are *(‘sofa’, ‘sofa’, ‘cushion’, ‘table’, ‘picture’)*

VI. Conclusion

We have described a framework for the object-goal navigation (ObjectNav) task, which requires a robot to find and move to an instance of a target object class in previously unseen scenes. The framework uses robot trajectories collected from other related scenes during a training phase to learn a Spatial Relational Graph (SRG) and Graph Convolutional Network (GCN)-based embeddings for the proximity of different semantically-labeled regions and the occurrence of different object classes in these regions. When the robot has to locate a target object instance during evaluation, Bayesian inference and the SRG are used to estimate the visible regions, and the GCN embeddings are used to rank and select the visible region to explore next. We have experimentally evaluated our framework using scenes from the Matterport3D (MP3D) benchmark dataset of indoor scenes in the visually realistic AI Habitat simulation environment. The quantitative and qualitative experimental results have demonstrated an improvement in the ability to locate the target object in comparison with baselines methods. Also, our framework significantly improves transparency while providing performance comparable with that of "black box", deep network-based approach for visible region estimation.

Future work will extend our framework by relaxing its assumptions. First, instead of assuming that the robot can accurately recognize the observed objects in any given image, we will enable the robot to perform probabilistic object recognition, potentially by also considering neighboring objects. Second, we will extract and use knowledge about indoor regions and objects from publicly-available knowledge graphs, and explore the performance of our framework on benchmark datasets of different origin (e.g., ReplicaCAD, Gibson). Finally, we will conduct trials on a physical robot to investigate the adaptation from simulation to the real world.

References

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

[2] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, “Matterport3d: Learning from rgb-d data in indoor environments,” *IEEE 3DV*, 2017.

[3] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, *et al.*, “Habitat: A platform for embodied ai research,” in *ICCV*, 2019, pp. 9339–9347.

[4] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, “Learning to explore using active neural slam,” in *ICLR*, 2020.

[5] S. K. Ramakrishnan, D. S. Chaplot, Z. Al-Halah, J. Malik, and K. Grauman, “Poni: Potential functions for objectgoal navigation with interaction-free learning,” *preprint arXiv:2201.10029*, 2022.

[6] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta, “Neural topological slam for visual navigation,” in *CVPR*, 2020.

[7] S. K. Ramakrishnan, Z. Al-Halah, and K. Grauman, “Occupancy anticipation for efficient exploration and navigation,” *arXiv*, vol. 2008.09285, 2020.

[8] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell, “Learning to navigate in complex environments,” *arXiv*, vol. 1611.03673, 2017.

[9] T. Chen, S. Gupta, and A. Gupta, “Learning exploration policies for navigation,” *arXiv*, vol. abs/1903.01959, 2019.

[10] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, “Target-driven visual navigation in indoor scenes using deep reinforcement learning,” *arXiv*, vol. 1609.05143, 2016.

[11] A. Mousavian, A. Toshev, M. Fiser, J. Kosecka, and J. Davidson, “Visual representations for semantic target driven navigation,” *arXiv*, vol. 1805.06066, 2018.

[12] Y. Wu, Y. Wu, A. Tamar, S. Russell, G. Gkioxari, and Y. Tian, “Bayesian relational memory for semantic visual navigation,” in *ICCV*, 2019, pp. 2769–2779.

[13] N. Savinov, A. Dosovitskiy, and V. Koltun, “Semi-parametric topological memory for navigation,” in *ICLR*, 2018.

[14] S. Zhang, X. Song, Y. Bai, W. Li, Y. Chu, and S. Jiang, “Hierarchical object-to-zone graph for object navigation,” *preprint arXiv:2109.02066*, 2021.

[15] N. Sünderhauf, “Where are the keys?–learning object-centric navigation policies on semantic maps with graph convolutional networks,” *preprint arXiv:1909.07376*, 2019.

[16] H. Du, X. Yu, and L. Zheng, “Learning object relation graph and tentative policy for visual navigation,” *arXiv*, vol. 2007.11018, 2020.

[17] Y. Qiu, A. Pal, and H. I. Christensen, “Learning hierarchical relationships for object-goal navigation,” *arXiv*, vol. 2003.06749, 2020.

[18] X. Hu, Z. Wu, K. Lv, S. Wang, and Y. Lin, “Agent-centric relation graph for object visual navigation,” *arXiv*, vol. abs/2111.14422, 2021.

[19] D. Batra, A. Gokaslan, A. Kembhavi, O. Maksymets, R. Mottaghi, M. Savva, A. Toshev, and E. Wijmans, “Objectnav revisited: On evaluation of embodied agents navigating to objects,” *preprint arXiv:2006.13171*, 2020.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *preprint arXiv:1301.3781*, 2013.

[21] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *preprint arXiv:1609.02907*, 2016.

[22] G. Kumar, N. S. Shankar, H. Didwania, R. Roychoudhury, B. Bhowmick, and K. M. Krishna, “Gcexp: Goal-conditioned exploration for object goal navigation,” in *RO-MAN*, 2021, pp. 123–130.

[23] D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov, “Object goal navigation using goal-oriented semantic exploration,” in *NeurIPS*, 2020.