Visual Object Search by Learning Spatial Context

Article *in* IEEE Robotics and Automation Letters · January 2020

DOI: 10.1109/LRA.2020.2967677

Asako Kanezaki
Tokyo Institute of Technology
48 PUBLICATIONS **1,149** CITATIONS

Visual Object Search by Learning Spatial Context

Raphael Druon[1,2], Yusuke Yoshiyasu[2,3], Asako Kanezaki[3], and Alassane Watt[2,4]

***Abstract*— We present a visual navigation approach that uses context information to navigate an agent to find and reach a target object. To learn context from the objects present in the scene, we transform visual information into an intermediate representation called context grid which essentially represents how much the object at the location is semantically similar to the target object. As this representation can encode the target object and other objects together, it allows us to navigate an agent in a human-inspired way: the agent will go to the likely place by seeing surrounding context objects in the beginning when the target is not visible and, once the target object comes into sight, it will reach the target quickly. Since context grid does not directly contain visual or semantic feature values that change according to introductions of new objects, such as new instances of the same object with different appearance or an object from a slightly different class, our navigation model generalizes well to unseen scenes/objects. Experimental results show that our approach outperforms previous approaches in navigating in unseen scenes, especially for broad scenes. We also evaluated human performances in the target-driven navigation task and compared with machine learning based navigation approaches including this work.**

I. Introduction

Target-driven visual navigation and object search [1], [2], [3] is a new approach that learns to navigate an agent in an environment based on the first-person visual information to reach a specific target location. Unlike previous navigation paradigms such as those using SLAM, this approach does not need a map of the environment.

One of the challenges in target-driven visual navigation is generalization to unknown scenes, since it relies on visual features coming from a scene image to produce actions, which is susceptible to appearance changes. In fact, most of the approaches in this field use a specific policy network for each scene [1], [3] or each scene type [2] e.g., kitchen or living room, and do not fully work well on unseen scenes. Hence, there is still a gap to use these navigation approaches in a wide range of different indoor scenes and it is also different from how humans generally navigate in such scenes. Another issue with the current navigation approaches is that their navigation ability degrades as the space to explore gets larger, because this increases the likelihood of the target object not being visible from the starting position.

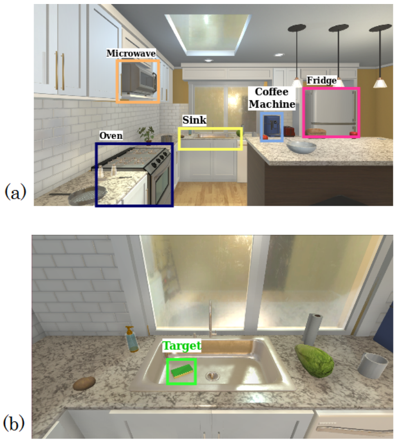


Fig. 1: An agent is at location (a) and looking for a sponge (b). From this location, the sponge is not visible. Other objects such as sink can easily be detected. Using these context objects as clue, e.g., a sponge should be near a sink, the agent can navigate to the target object even if the agent starts from the position where the object is not visible.

One way to improve generalization is to use semantic and contextual information to abstract visual information. The use of context has been explored in many fields and it plays an important role in many tasks, such as image retrieval [4], 3D scene synthesis [5] and object functionality analysis [6]. However, the effect of using context in navigation has not been explored until recently and requires further investigations.

Our approach is inspired by human behaviors for object searching. Imagine a situation where we look for an object in everyday life. To find the object, we will not look everywhere in the room but look at the most likely place we can find it. While doing this, we see the environment around us and other objects existing in our view, namely context, and use this as clues. Once the object of target comes into our sight, we will walk straight forward toward it. We want to mimic such behaviors by our agent. For example, the agent is looking for a sponge but it cannot be seen from Fig. 1 (a). Using context, the agent searches in the direction of the most probable location based on other visible objects, here the sink, and find and reach the target, here the sponge, after moving to the position where it is visible (Fig. 1 (b)).

In this paper, we propose a visual navigation approach to search for an object in indoor environments based on context information. To that end, we introduce an intermediate representation called *Context Grid* constructed from the bounding boxes of objects existing in the view. Contextual features are then extracted from this representation and used

[1] Raphael Druon is with Paul Sabatier University, France raphdruon@gmail.com

[2] Yusuke Yoshiyasu, Raphael Druon, and Alassane Watt are with CNRS-AIST JRL (Joint Robotics Laboratory) UMI3218/RL yusuke-yoshiyasu@aist.go.jp

[3] Yusuke Yoshiyasu and Asako Kanezaki are with National Institute of Advanced Industrial Science and Technology (AIST), Japan kanezaki.asako@aist.go.jp

[4] Alassane Watt is with CentraleSupelec, France wattalassane@gmail.com

by an agent to navigate the scene by learning a mapping from the contextual features to its actions. Since the context grid contains information about both the target object and context objects present in the view, this representation thus enables the agent to navigate not only quickly to the target when it's visible but also to the likely location even when the target is not visible based on context. This improves the ability of the agent to find hidden objects or small objects that are difficult to see from the starting point. Since this representation is robust against appearance changes, our approach can generalize to unseen scenes of different scene types using a single policy network. We evaluate our method with various synthetic 3D scenes and compare navigation performance against previous approaches as well as human subjects.

II. Related work

Context reasoning and learning Understanding and exploiting context information of 2D/3D scenes is an important problem in the computer vision and computer graphics field, which has been explored extensively and playing an important role in many tasks. Fisher et al. [5] proposed 3D scene retrieval and synthesis systems that exploit contextual information to arrange 3D models in a plausible way. Hu et al. [6] introduced a contextual descriptor called ICON which aims to provide a geometric description of the functionality of a 3D object in the context of a given scene. Context has also been used in the basic computer vision tasks such as object detection [7]. Johnson et al. [4] proposed a richer context representation called scene graphs where relationships between objects, such as spatial, comparative, verb, prepositional, are represented by graph structures. Using scene graphs, more complicated tasks like image captioning and image generation [8] can be done.

Map-based navigation approaches Classical navigation methods [9], [10] use prior information of their environments based on a map, which is often computed on the fly using SLAM techniques. Once the environment map is created, global path planning algorithms such as A* search [11], probabilistic roadmaps (PRM) [12], and rapidly-exploring random trees (RRT) [13] can be applied to generate way-points that navigate an agent from the start to the goal location. Recently, deep learning based approaches [14], [15] are proposed to substitute the traditional path planning methods to make the navigation more robust to dynamic scenes. Deep reinforcement learning methods are also proposed to learn obstacle avoidance behavior and used in the map-based navigation framework [16], [17], [18]. In those works, the target location of the navigation is given. Dong et al. [19] proposed an online path planning algorithm for multiple robots to achieve 3D reconstruction of unknown indoor environments, where no target location is given but the reconstruction quality is maximized and the overall scanning effort is minimized.

Map-less navigation techniques In map-less navigation approaches, no map is needed to navigate a robot in its environment. Chen et al. [20] learned to predict the affordance for driving actions from driving images using deep convolutional neural network framework. Gupta et al. [21] proposed to navigate in broad indoor environments using visual information and egomotion, which learns to construct a latent map and navigates in it using deep learning. Giusti et al. [22] proposed to learn a policy for a Micro Aerial Vehicle (MAV) to follow a trail using visual information in an outdoor environment, while Sadeghi and Levin [23] learned control policies for a MAV using reinforcement learning in a simulated CAD environment. The objective of these works is to learn navigation affordance from visual features where no target object is explicitly given.

Target-driven navigation techniques Target-driven visual navigation approaches [3], [1], [2] use the first-person visual information from the scene to navigate an agent in it. Zhu et al. [3] proposed the first method in this field, which uses an image of the goal scene as target to navigate an agent. Wortsman et al. [24] recently proposed a meta-learning approach to target-driven navigation to obtain the ability to learn from its mistake during the test time. Ye et al. [1] proposed a method that instead searches for an object of interest. Their focus was to quickly find a specific instance of object shown as image by designing a customized object recognition module that localizes an object in a whole image. While they validated their method by finding 4 different objects in a scene, they did not consider generalizations to unseen objects or different instances of the same object.

Concurrently, Yang et al. [2] propose to extract object relation features by fusing ResNet and word embedding features through Graph Convolutional Network (GCN) and abstract the goal using a semantic representation of the target name. In contrast, we propose to use spatial context information established from detection of both, the target and other objects, in the current view using a well-established object detector [25] and feed this to the network. As we will show later, this makes our navigation model robust against appearance changes and generalize better to unknown scenes/objects. In fact, our approach uses a single policy network for all of the scene types, whereas [2] uses a different policy for each scene type (e.g. kitchen, bath room) and has some difficulties in generalizing to unseen scenes.

III. Target-driven visual navigation model

A. Task definition

Given a target object, e.g. a pen, our task is to navigate an agent to an instance of this object category in a scene. We consider a set of scenes $S = \{s_0, \ldots, s_n\}$ and target objects $O = \{o_0, \ldots, o_m\}$. A task is the combination of a scene s_i, target object o_i and the initial position p_i of the agent, which is denoted as $T = \{s_i, o_i, p_i\}$. The agent starts at position p_i in scene s_i and try to reach a state where object o_i is visible. More precisely, the goal is reached only if the object is visible, i.e. the object is visible if it is present in the current view and located within 1 meter from the agent [26]. For each current state, the agent will take an action a from a set of actions, A = $\{MoveAhead$, $MoveBack$,

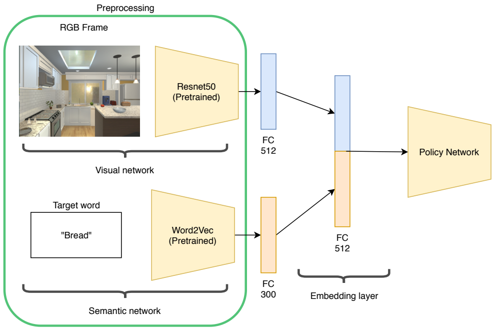


Fig. 2: Baseline network architecture

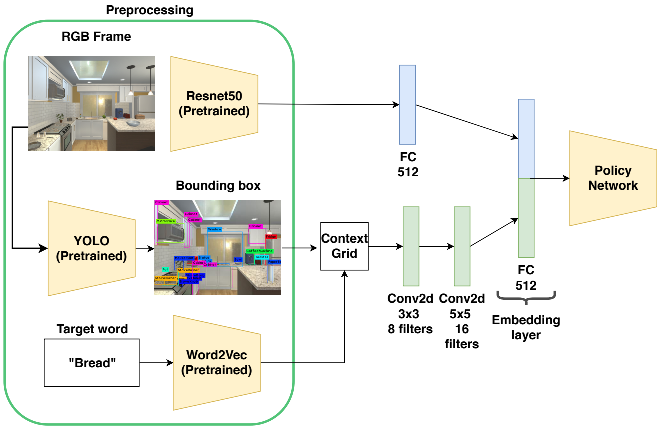


Fig. 3: Our network architecture. Context grid is obtained by combining bounding box information from YOLO [25] and cosine similarity between target and detected object names. We show an example of YOLO bounding boxes and context grid in Figure 4.

$MoveRight$, $MoveLeft$, $RotateRight$, $RotateLeft$, $LookUp$, $LookDown$, $DONE\}$, resulting in a new state unless a collision occurs. Each action is defined as follows:

- $MoveAhead$, $MoveBack$, $MoveRight$, $MoveLeft$ will move the agent respectively forward/backward/to the right/to the left.
- $RotateRight$, $RotateLeft$ will rotate the agent respectively to the right/left 45 degrees.
- $LookUp$, $LookDown$ will tilt the camera respectively up/down with 30 degrees.
- $DONE$ is a special termination action.

DONE signal This special action will indicate to the environment that the agent is ready for evaluation. *Success* is denoted by a $DONE$ signal produced by the agent in a state where the goal is visible. If the goal is not visible and agent produces a $DONE$ signal, it represents a *failure*. The visibility definition allows us to avoid explicitly supervising the exact positions of the target objects. As a result, it can work in a real scenario where the position of the goal is not necessarily known. Note that as also mentioned in [2] the use of $DONE$ signal makes the navigation problem more challenging because the agent needs to learn to stop at the goal by itself.

B. Baseline model

We use a deep reinforcement learning framework for learning navigation using a set of visual and semantic inputs to find and reach the target object. As a baseline model, we use a network architecture that outputs an action from a semantic target and visual information inputs (Fig. 2), which partially shares common network structures as recent approaches [24], [2]. Note that in our work a single policy network is used for all the scenes. Given a semantic target $\boldsymbol{g} \in \mathbb{R}^{300}$ and visual feature $\boldsymbol{v}_s \in \mathbb{R}^{512}$ for each state s at time step t, the policy network will output an action a_t for the current time step t. The policy function can be expressed as follows:

$$a_t \approx \pi(f(\boldsymbol{g}; \boldsymbol{v}_s; \boldsymbol{\theta}))$$

where f is the neural network and $\boldsymbol{\theta}$ is its parameters.

Visual network We use the same visual feature extraction mechanism as in [3] where the current view of the agent is given to the ResNet50 [27] network to extract features from the RGB frame. The visual network is pre-trained on ImageNet and frozen during navigation training. We remove the last fully-connected layer of the network, i.e., the layer just after the last average pooling. This results in a 2048-d feature for an input of 400×300. To account for the history, the last four previous frames are stacked together, resulting in a 8192-d feature. These features are then passed into a fully-connected layer with ReLU activation that outputs a 512-d feature.

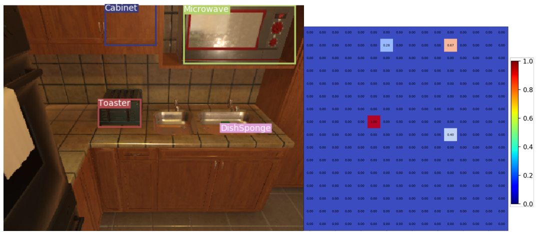
Semantic network To extract features from the goal object represented as the text of the object class, we use Spacy [28] toolkit to extract word-embedding, resulting in 300-d feature per class word. This 300-d feature is then passed into a fully connected layer with ReLU activation that outputs 300-d feature. Using word embedding as input features, it will help the network to generalize to unseen goals which is semantically close to the target objects used in training e.g. a fork will be close to a knife.

Embedding These two types of features are then stacked together to produce a joint embedding layer. As semantic and visual features are expressed in the different modality spaces, this layer will help the network to generalize to a new scene and goal. This joint embedding layer is a fully-connected layer that outputs 512-d features that are passed to the actor critic model.

Actor-Critic policy network We use the Asynchronous Advantage Actor-Critic model (A3C) [29] to predict the policy and the value at each step. The input to this network is the output features from the joint embedding layer and it outputs two values: the next action (output size is $|A|$) and the Q-value for the current state. The network contains two fully-connected layers.

C. Reward

We use a mixture of reward functions proposed by [3] and [1]. We want to minimize the path length between the start



(a) YOLO output (b) Context grid (16 × 16)

Fig. 4: Example of YOLO output bounding boxes (a) and context grid produced (b). The target object is *Toaster.*

position and the goal. We also want to maximize the area of the bounding box of the target class. As stated in III-A, we use the special signal $DONE$ to force the agent to learn the task. To do this we propose the novel reward function:

$$Reward = \begin{cases} 5, & \text{if } DONE \text{ signal is emitted and object is visible} \\ S_{\text{bbox}} & \text{if } S_{\text{bbox}} \text{ is the highest in the episode} \\ -0.01, & \text{otherwise} \end{cases} \quad (1)$$

where S_{bbox} is the bounding box area normalized by the total area view. For example, $S_{\text{bbox}} = 1$ means the bounding box covers the entire agent's field of view. The idea behind this reward is to penalize an action that does not lead to a state where the object is more visible than the previous state, which means that the agent is not getting closer to the object.

This reward function will first force the network to keep getting closer to the goal using the bounding box area reward, even if the state is not the final state. Then it will force the network to output the $DONE$ signal only when the goal is visible. This has the largest impact for the network to learn to navigate to the goal, as the $DONE$ signal determines if the episode is success or failure during the evaluation.

IV. Spatial context network for object search

Our network model for object searching using context information is shown in Fig. 3. To learn and use context in navigation, we propose to use an intermediate representation which we refer to as context grid. To extract context features, the context grid is passed to the context network that consists of two 2D convolution layers with ReLU activation and followed by max pooling. Unlike the baseline model in Fig. 2, we also propose to remove the semantic network which uses a semantic target as input, as the context grid already encodes this information.

The context grid is constructed from the bounding boxes of both the target object and context objects present in the view. This allows us to navigate the agent not only quickly to the target when it's visible but also to the mostly likely location based on context information even when the target is not visible. The size of the context grid used in this paper is 16×16 and an example of context grid is shown in Fig. 4. The context grid uses the detected class of the object and computes the similarity between the target object and the viewed object. The position corresponding to the center of the bounding box is filled with similarity values between the target object and the object at that location. For example, if the target object is "bottle" and a bottle is detected by the object detection network, a value of 1 is assigned at the location of the bottle in the current view of the agent. If a glass is detected, the similarity score between "glass" and "bottle" is assigned at the glass location. Otherwise, 0 is assigned.

We use a well-established object detector YOLOv3 [25] to detect objects in the current view. To compute the semantic similarity between the target and detected object, we use the word-embedding feature $\boldsymbol{v}$ from the detected class x and the target class y in the same manner as the semantic network (Fig. 2). We then compute the cosine similarity (CS) between these two vectors using the following formula:

$$CS(\boldsymbol{v}_x, \boldsymbol{v}_y) = \frac{\boldsymbol{v}_x \cdot \boldsymbol{v}_y}{||\boldsymbol{v}_x|| \cdot ||\boldsymbol{v}_y||}$$

If two objects overlap in the context grid, we take the maximum between the two similarity values. Here the maximum is used because a higher value indicates an object more similar to the target. To further capture spatial relationship rather than the semantic relationship, we also propose to extract word embedding features trained on visual genome dataset [30]. To do so, we used the word2vec [31] implementation available in gensim [32] to train the model on visual genome region captions. This allows us to extract more meaningful real-world object relationships in images.

The main motivation behind this novel approach using context grid is to improve its navigation capability based on a human-inspired object searching strategy. As we construct context grid from object detection results, when the target is visible and detected, the agent will be strongly pushed towards the target object by our network. Even if the target object is not visible, the agent will be moved to the location where the target object likely exists because the context grid also contains the similarities of other objects which are semantically similar to the target.

Another motivation is generalization to unknown scenes/targets. By providing the context grid representation which encodes semantic relations between objects to the network, the agent will first learn to use this representation to navigate in the training environment and can then use it in an unseen environment in the same manner. This is because we construct context grid from scalar cosine similarities which are already transformed and compressed from 300-d word-embedding vectors that will likely change when we introduce semantically new objects. With context grid, an unseen target is handled in a similar way as the ones seen in training and carries the same semantic information even during testing. Thus, the spatial context network will not be disrupted by unseen target inputs and can generalize to unseen scenes and target objects.

The context grid is a simple yet effective representation that encodes semantics and locations of objects. Our context grid is different from the scene layout [8] as it encodes semantic similarities of the objects w.r.t. the target, which is the key to generalization to unseen scenes. A more sophisticated way to represent context could be to use scene graphs [4] that represent a more detailed relationships. Yang et al. [2] used GCN to encode object relationship, which is in spirit similar to scene graphs but they could not explicitly use detailed relationship such as top, bottom, left, right, because of the scarcity in these relation labels. We believe that for a large-scale problem like navigation a simple representation like context grid is sufficient and easier to learn features from it. In fact, our approach outperforms GCN by a large margin in unseen environment settings.

V. Experiments

A. Dataset

The dataset used in this paper is AI2-THOR for (The House Of inteRactions) [33]. This framework proposes 120 photo-realistic environments divide into 4 different room types (Bathroom, Living room, Kitchen and Bedroom) with 30 different environments per room.

For our experiment, we split the dataset by following [26]. The training set is composed of the first 10 scenes out of 20 from each room type. The test set is composed of 5 scenes. For both training and testing sets, we use the same goal to compare our approach against the other agent models without context grid. The goals used in the experiment are specific to each room type and are as follows : Kitchen: *Toaster, Microwave, Fridge, CoffeeMachine, GarbageCan, Bowl*; Living room: *Pillow, Laptop, Television, GarbageCan, Bowl*; Bedroom: *HousePlant, Lamp, Book, AlarmClock*; Bathroom: *Sink, ToiletPaper, SoapBottle, LightSwitch*. The object positions are determined by the AI2-THOR framework to a randomly but likely positions. We also evaluate the generalization of our approach to unseen goals. To do so we compute the closest objects in the word embedding space for each object presents in the train and get the most similar object for the evaluation. Goals produced are as follows: Kitchen: *Mug, Pot, Cup*; Living room: *Sofa, Box, TableTop*; Bedroom: *Mirror, CD, CellPhone*; Bathroom: *Toilet, Towel*. Note that some objects are not present or visible in the scene and are removed from the possible goal for this specific scene.

We discretize the environments using 0.5 meters steps, i.e, each position will be 0.5 meters apart from each other. According to the action set in Table III-A, there is eight possible views for every position in the environment, resulting in an average of approximately 500 possible states in each scene.

B. Implementations

We implement our framework using PyTorch, partially based on a publicly-available implementation [34] of the original target-driven visual navigation technique [3]. To speed up the training process, we precompute visual features for every frame in every scene using ResNet50. We trained YOLOv3 [25] network using the ground-truth bounding boxes extracted from the train scenes of AI2-Thor dataset. We also precomputed the word embedding of every object class present in all the scenes. We trained YOLOv3 using 147 classes (total number of available objects on the AI2-THOR framework), the training dataset contains 104 of these classes, the evaluation set contains 91 different classes.

We use the RMSProp optimizer with a learning rate of $7e-4$ and decrease it linearly until 0 in the last epoch. We use the same hyper-parameters, epochs and reward function for every implementation, except for the random, in order to perform meaningful comparisons. The network is trained for 24 hours, with a total processing of 25 million frames using Nvidia TITAN V GPU. For the evaluation we use the test set presented previously. If no signal is emitted by the agent to tell that the object is found, the episode stops with 300 steps, which is a failure. We run the evaluation on all test scenes during 250 episodes per scene/goal.

C. Metrics

In this paper we use the same metrics presented by [26], which is used by [24], [2]. This metric is called Success weighted by normalized inverse Path Length (SPL). SPL is calculated as follows:

$$\text{SPL} = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{p_i} \tag{2}$$

where N is the number of evaluation's episode. S_i is 1 if the episode is a success, else 0. l_i is the shortest path length between the start position and one of the success states. p_i is the length of the current episode. The length used here is the number of action, meaning that taking 1 action increments the length by 1. This metric provides a balance between episode length and the success rates.

Using the above notation we also evaluate the success rate as follows:

$$\text{Success rate} = \frac{1}{N}\sum_{i=1}^{N} S_i \tag{3}$$

D. Agent models

Here we describe the agent models evaluated and compared in the experiment. In this paper, one unique model is used across all scenes, contrary to [2] which uses one model per room type. There are six models: (1) **Random**, (2) **Baseline**, (3) **GCN**, (4) **Human** and (5) **Ours**.

Random corresponds to the most simple navigation agent that randomly samples an action from the set of actions A. **Baseline** corresponds to the baseline model presented in Section III-B. **GCN** corresponds to our implementation of [2] based on the model implemented by [24]. **Human** is human subjects. Please see Appendix for more information on the experimental protocols and results. **Ours** is the our model that uses context grid created from multiple bounding boxes including the target, which are detected by YOLOv3 [25]. **Ours (w/o context)** is our model using only the bounding box of the target object. It's similar to **Ours**, having 1 at the target object, if it presents in the view, but no similarity value for other object classes. **Ours (GT)** corresponds to the approach presented in this paper using the ground truth bounding boxes. **Ours (LSTM_3layers)** corresponds to the same approach as **Ours (GT)**, but we replaced the 4 stacked frame as input to a single frame and we added 3 stacked LSTM layer after the embedding layer (Blue/Green layer in Fig. 3) to capture longer-term temporal information. **Ours (spatial)** is our model using context grid filled with spatial similarities from visual genome dataset [30].

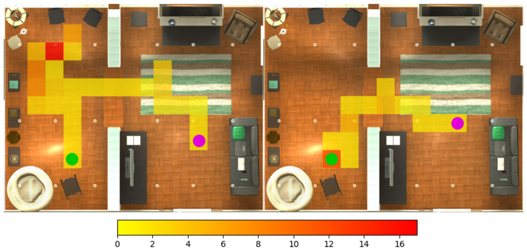
E. Results and discussion

Comparison to baseline and state-of-the-art Table I shows the evaluation results using $DONE$ signals. For known scenes and objects, all of the agent models, except for random, works well. For unseen objects and scenes, we can see that the best results are obtained using an object detector (Ours and Ours (w/o context)), which outperforms GCN and the baseline with large margins. Without using the $DONE$ signal (Table II), the success rate of ours reaches up to around 70%, as in this easier setting the environment stops the agent automatically when it reaches the target.

As with our approach, GCN [2] extracts object relations using their relationship graphs. However, this property is not well exploited to generalize to unseen scene and object settings, which is evident from the results shown in the original paper (Table 1 of [2]) and obtained using our implementation. One reason for this is that, in GCN [2] and previous approaches such as [1], the networks need to learn the representation of the objects to locate the target object using the ResNet feature whose spatial resolution are already lost. Using the object detector as in our method can remove this learning step and makes the object searching problem easier. Another reason is GCN [2] learns mappings from visual features from ResNet and semantic features from word embedding but they can change easily by scene appearance and introductions of new objects, even though the scene is semantically similar.

We tried different variants of recurrent neural network (RNN cell, GRU cell, 2 and 3 layers LSTM) that is commonly used to learn temporal information. The best recurrent model in terms of SPL is the 3 layers LSTM, which did not however reach the level of the non recurrent version (other results are available in the appendix). This is probably because the space to explore is not very large in the AI-2Thor framework, since each scene basically consists of a single room, and longer-term temporal information is not well exploited. For a scene with multiple rooms, these recurrent models may work better than the method that stacks features.

We also compared trajectories between GCN [2] and our method. The GCN trajectory took longer steps than ours and got stuck, colliding with the environment (as shown in the red square in Fig. 5 (a)). Our method searches for the object in the beginning by looking around (as shown in the orange square in the Fig. 5 (b)) and then proceed to navigate toward the goal without colliding with the environment. This behavior can be explained by the fact that our method uses bounding box information to navigate the agent and the agent will reach the goal as soon as it finds the target object in the view, as shown Fig. 6.



(a) GCN[2] trajectory (b) Our method trajectory

Fig. 5: Comparison of navigation trajectories for GCN and our method. The task is to find a laptop on a sofa. Green dot is the start state. Purple dot is a goal state. Colored squares indicate the number of actions taken at each position. GCN takes longer steps than ours.

Generalization to new scene types To evaluate the ability of generalization to new scene types, we trained our network on two scene types (Kitchen and Bedroom) and evaluate it on the two others (Living room and Bathroom) under the unknown scenes & unknown objects setting. Note that the results were similar for unknown scenes & known objects evaluation. While the results are not as good as the ones obtained with our models trained with all the scene types, it achieves the success rate of 35% and SPL of 9 and outperforms baselines and GCN. Increasing the training pool, by adding the Living room, improves the success rate to approx. 45% and SPL to 12. Evaluation without the $DONE$ signal gives better results as expected with the success rate of approx. 70% and a SPL of 35. With a increased training pool, success rate is at approx. 80% with a SPL of 45.

Robustness against space discretization To investigate the generalization ability of our model to go beyond strict space discretization, we evaluated our model, which is trained with 0.5 meter steps, on the 0.25 and 0.1 meter step space discretization, which are closer to real scenarios. The results of these evaluations are available in Table I. Even though the model is trained in 0.5 meters step, it is able to generalize to smaller step sizes. We believe that our method can also be applied to continuous space using controls from a physics engine as done in [3].

Impact of context and object detection quality **Ours** (using context) generally yields better results than **Ours (w/o context)** that uses only the target object position. **Ours (w/o context)** however can perform well in small scenes (Bathroom). This can be explained by the fact that the target objects are visible from the start positions in small scenes more frequently than broader scenes and thus the network using the target bounding box solely achieves a good performance in this case.

We then compare the results by replacing the bounding boxes to the ground truth. The results are slightly better using the ground truth bounding boxes but the improvements are minor. YOLOv3 does not detect all the objects in the scenes but most of them are found. The mAP@IoU=50% score of YOLO is 40.49% and its F1-score is 0.7. Even if mAP is low,

TABLE I: Evaluation results using the $DONE$ signal. SPL and Success rate are in percentage.

Eval. set	Method	Kitchen		Living room		Bedroom		Bathroom		**Average**	
		SPL	Success	SPL	Success	SPL	Success	SPL	Success	SPL	Success
Known scenes & Known objects	Random	0.67	4.82	0.49	2.6	0.66	3.38	1.14	7.79	0.74	4.65
	Baseline	57	99.367	54.1	89.82	66.2	99.253	52.3	99.2	57.4	96.91
	GCN [2]	55.05	97.781	51.5	86.54	65.74	97.734	53.79	95.55	56.52	94.40
	Ours (w/o context)	57.87	96.48	50.12	86.95	62.71	99.02	53.68	99.44	56.09	95.47
	Ours (GT)	57.7	98.979	51.3	86.16	64.4	98.5	53	99.32	56.6	95.74
Unknown scenes & Known objects	Random	1.08	8.41	0.74	4.14	0.62	3.32	1.28	8.92	0.93	6.2
	Human	26.676	91.866	28.726	89	33.13	96.666	24.024	97	28.139	93.633
	Baseline	4	18.314	3.8	19.68	0.5	4.33	5	26.32	3.325	17.16
	GCN [2]	5.38	19.668	3.2	15	1.52	7.506	7.14	27.2	4.31	17.35
	Ours	17.56	61.314	12.8	44.84	8.04	30.12	15.02	56.58	13.355	48.21
	Ours (spatial)	18.28	62.866	12.62	44.42	7.66	29.718	15.52	56.52	13.52	48.381
	Ours (w/o context)	16.32	53.418	9.4	39.62	6.86	28.292	13.92	55.04	11.625	44.1
	Ours (GT)	20	58.488	17.6	49.9	**17.68**	48.986	20.34	66.06	18.91	55.86
	Ours (LSTM_3layers)	19.1	51.994	17.36	44.34	7.78	25.134	17.42	59.82	15.415	45.322
	Ours (GT) @ 0.25m	15.88	53.274	13.86	42.9	7.08	28.372	17.22	56.96	13.51	45.3765
	Ours (GT) @ 0.1m	13.72	46.498	11.2	37.85	6.9	23.812	15.56	51.24	11.845	39.85
	Ours (GT & w/o context)	**22.12**	**64.53**	13.32	39.68	11.28	38.32	**22.34**	**71.16**	17.27	53.42
	Ours (GT & spatial)	21.18	59.978	**18.26**	**59.98**	16.76	**49.854**	20.38	64.3	**19.145**	**58.53**
Unknown scenes & Unknown objects	Baseline	5.34	18.14	3.16	18	1.925	8.75	1.64	9.414	3.02	13.58
	GCN [2]	5.1	16.36	2.4	14.52	1.725	6.115	1.84	11.814	2.77	12.2
	Ours	4.7	31.068	12.38	37.026	3.82	21.626	10.28	49.48	7.79	34.80
	Ours (spatial)	5.7	33.52	13.02	38.308	3.54	20.024	14.36	67.92	9.15	39.943
	Ours (w/o context)	4.7	24.132	2.76	14.626	3.92	18.374	15.24	72.68	6.655	32.453
	Ours (GT)	12.42	52.374	15.76	46.548	**12.56**	**42.934**	11.7	50.04	13.11	47.97
	Ours (LSTM_3layers)	15.5	49.882	13.3	45.774	10.16	31.386	11.68	37.4	12.66	41.115
	Ours (GT) @ 0.25m	8.36	42.534	10.74	40.158	10.94	38.134	9.38	32.64	9.855	38.3665
	Ours (GT) @ 0.1m	10.8	42.534	10.475	41.4325	8.28	27.334	9.42	34.8	9.74375	36.525125
	Ours (GT & w/o context)	**17.26**	46.266	3.56	20.492	9.18	30.586	**20.14**	**72.72**	12.54	42.52
	Ours (GT & spatial)	12.6	**55.492**	**17.76**	**50.5**	11.08	37.654	17.76	59.6	**14.80**	**50.81**

TABLE II: Evaluation results without the $DONE$ signal (the agent is automatically stopped when it reaches the target).

Evaluation set	Method	Kitchen		Living room		Bedroom		Bathroom		**Average**	
		SPL [%]	Success [%]	SPL	Success	SPL	Success	SPL	Success	SPL	Success
Unknown scenes, Known objects	GCN [2]	27.32	59.65	12.58	34.72	10.48	40.052	32.58	68.74	20.74	50.7905
	Ours	42.74	81.152	28.92	66.24	25.14	63.534	44.56	74.68	35.34	71.4015
	Ours (GT)	45.62	78.6	33.72	69.92	35.08	63.812	46.26	74.94	40.17	71.818
	Ours (LSTM_3layers)	45.2	78.58	32.92	72.34	19.94	44.30	45.52	74.7	35.9	67.48
Unknown scenes, Unknown objects	GCN [2]	27.9	59.104	12.6	34.36	10.82	39.986	32.36	67.86	20.92	50.3275
	Ours	25.42	70.426	34.96	62.694	14.62	51.572	40.78	75.08	28.945	64.943
	Ours (GT)	39.92	81.172	35.8	64.478	29.54	59.386	43	72.72	37.065	69.439
	Ours (LSTM_3layers)	43.28	80.6	34.82	68.81	24.98	57.36	45.56	73.76	37.16	70.13

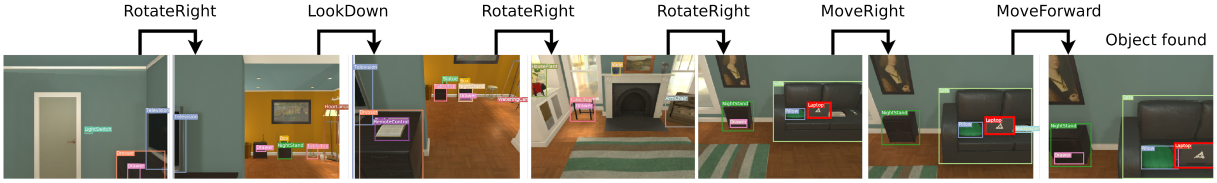


Fig. 6: The sequence of the agent's views while navigating in the scene of Fig. 5 using our model. As soon as the target object (laptop) is found, the agent reaches the goal. The bounding box of the target object is in thick red.

this does not cause major problems for our application as we do not need precise bounding box locations. The detection results could be improved using image augmentations or adding real images to the train set.

RL agents vs human From the human evaluation of the same navigation task, we can see that human subjects obtain almost perfect scores in terms of success rate, even in an unknown environment. In contrast, the SPL scores of human subjects are not as good as their success rates, which is almost the same as our RL agent. This implies that humans are good at exploring a new environment and once they detects the target object they will almost always be able to get to the goal in the end. However, humans tends to search for objects and does not take the shortest path to reach it.

Failure cases Failures can be split into two cases: 1) *lost failure* where the agent exceeds the time limit (300 steps) and is stopped by the environment; 2) *done failure* where the agent produces the $DONE$ signal at wrong locations. Based on our results, about half of the failures are *lost failures* and the other half are *done failures*.

In most of *lost failures*, the agent stuck in the same space by repeating the same actions (e.g., repeat left and right movements, rotating around right or left at same place), which does not change the state of the agent. This behaviour may come from an over-fitting of the model as the next action is sampled from a multinomial distribution or the lack of sensory input such as collision information. The 3-layered LSTM model decreases the *lost failure* rate from ≈ 40 % to ≈ 20 %, showing the capability of this model to capture temporal information which can be used to avoid repeating the same action and being stuck at the same place.

In most of *done failures*, the target object is present in the current view of the environment but the agent is too far from it, meaning the agent understands the target object but not about its depth. We believe that using depth information from a depth sensor or laser scanner can help the model overcome this type of failure. Also, the agent sometimes confuses objects with similar semantics and function (e.g. *Towel* with *Curtains*, *Cup* with *Bowl* and *Mirror* with *Window*).

VI. Conclusion

We presented a novel visual navigation approach that learns and exploits spatial context information in a scene to find an object. To do so we proposed an intermediate representation called context grid which is constructed from the bounding boxes of objects present in the scene and their semantic features from word embedding. We show that the use of this representation in navigation provides a clear advantage for generalization to unseen scenes and goals.

We also evaluated human performance on the same navigation task and compared against our approach as well as RL approaches. Unlike image classification and game play such as Go, the performance of RL navigation agents still does not reach the human level. This is probably due to the fact that in the current target-driven navigation approaches the agent does not have access to enough information from the environment such as depth or collision feedback. It would be interesting to add such information in our navigation framework. Also, combining with structured domain randomization [35] would be a promising way to use our method in real indoor scenes. Other interesting future directions would be to extend the framework to handle a bigger space like a whole house, more complicated tasks like finding a list of objects in a supermarket, complex commands provided as a sentence and so on.

References

[1] X. Ye, Z. Lin, H. Li, S. Zheng, and Y. Yang, "Active object perceiver: Recognition-guided policy learning for object searching on mobile robots," in *IROS*, 2018.

[2] W. Yang, X. Wang, A. Farhadi, A. Gupta, and R. Mottaghi, "Visual semantic navigation using scene priors," in *ICLR*, 2019.

[3] 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," in *ICRA*, 2017.

[4] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei, "Image retrieval using scene graphs," in *CVPR*, 2015.

[5] M. Fisher, D. Ritchie, M. Savva, T. Funkhouser, and P. Hanrahan, "Example-based synthesis of 3d object arrangements," in *ACM SIGGRAPH Asia*, 2012.

[6] R. Hu, C. Zhu, O. van Kaick, L. Liu, A. Shamir, and H. Zhang, "Interaction context (ICON): towards a geometric functionality descriptor," *ACM Transactions on Graphics*, vol. 34, no. 4, pp. 1–12, 2015.

[7] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The Role of Context for Object Detection and Semantic Segmentation in the Wild," in *CVPR*, 2014.

[8] J. Johnson, A. Gupta, and L. Fei-Fei, "Image Generation from Scene Graphs," in *CVPR*, 2018.

[9] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *Autonomous robot vehicles*, pp. 396–404, 1986.

[10] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278–288, 1991.

[11] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[12] L. E. Kavralu, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robotics and Automation*, vol. 12, no. 4, 1996.

[13] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[14] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, "Value iteration networks," in *NIPS*, 2016.

[15] A. Kanezaki, J. Nitta, and Y. Sasaki, "Goselo: Goal-directed obstacle and self-location map for robot navigation using reactive neural networks," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 696–703, 2018.

[16] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, *et al.*, "Learning to navigate in complex environments," in *ICLR*, 2017.

[17] A. Faust, K. Oslund, O. Ramirez, A. Francis, L. Tapia, M. Fiser, and J. Davidson, "Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning," in *ICRA*, 2018.

[18] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end-to-end with autorl," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2007–2014, 2019.

[19] S. Dong, K. Xu, Q. Zhou, A. Tagliasacchi, S. Xin, M. Nießner, and B. Chen, "Multi-robot collaborative dense scene reconstruction," *ACM Transactions on Graphics*, vol. 38, no. 4, p. Article 84, 2019.

[20] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *ICCV*, 2015.

[21] S. Gupta, V. Tolani, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive Mapping and Planning for Visual Navigation," in *CVPR*, 2017.

[22] A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, D. Scaramuzza, and L. M. Gambardella, "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2016.

[23] F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," in *Robotics: Science and Systems*, 2017.

[24] M. Wortsman, K. Ehsani, M. Rastegari, A. Farhadi, and R. Mottaghi, "Learning to Learn How to Learn: Self-Adaptive Visual Navigation Using Meta-Learning," in *CVPR*, 2019.

[25] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[26] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. R. Zamir, "On Evaluation of Embodied Navigation Agents," *arXiv preprint arXiv:1807.06757*, 2018.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *CVPR*, 2016.

[28] M. Honnibal and I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing," 2017, to appear.

[29] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," in *ICML*, 2016.

[30] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei, "Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations," *Int. J. Comput. Vision*, vol. 123, no. 1, pp. 32–73, 2017.

[31] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," in *ICLR Workshop*, 2013.

[32] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *LREC Workshop on New Challenges for NLP Frameworks*, 2010.

[33] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "AI2-THOR: An Interactive 3D Environment for Visual AI," *arXiv preprint arXiv:1712.05474*, 2017.

[34] J. Kulhnek, "Visual navigation pytorch implementation," https://github.com/jkulhanek/visual-navigation-agent-pytorch.

[35] A. Prakash, S. Boochoon, M. Brophy, D. Acuna, E. Cameracci, G. State, O. Shapira, and S. Birchfield, "Structured domain randomization: Bridging the reality gap by context-aware synthetic data," in *ICRA*, 2019, pp. 7249–7255.