Spot the Difference: A Novel Task for Embodied Agents in Changing Environments

Federico Landi, Roberto Bigazzi, Marcella Cornia, Silvia Cascianelli, Lorenzo Baraldi, Rita Cucchiara
University of Modena and Reggio Emilia
Email: {name.surname}@unimore.it

Abstract—Embodied AI is a recent research area that aims at creating intelligent agents that can move and operate inside an environment. Existing approaches in this field demand the agents to act in completely new and unexplored scenes. However, this setting is far from realistic use cases that instead require executing multiple tasks in the same environment. Even if the environment changes over time, the agent could still count on its global knowledge about the scene while trying to adapt its internal representation to the current state of the environment. To make a step towards this setting, we propose Spot the Difference: a novel task for Embodied AI where the agent has access to an outdated map of the environment and needs to recover the correct layout in a fixed time budget. To this end, we collect a new dataset of occupancy maps starting from existing datasets of 3D spaces and generating a number of possible layouts for a single environment. This dataset can be employed in the popular Habitat simulator and is fully compliant with existing methods that employ reconstructed occupancy maps during navigation. Furthermore, we propose an exploration policy that can take advantage of previous knowledge of the environment and identify changes in the scene faster and more effectively than existing agents. Experimental results show that the proposed architecture outperforms existing state-of-the-art models for exploration on this new setting.

I. INTRODUCTION

Imagine you have just bought a personal robot, and you ask it to bring you a cup of tea. It will start roaming around the house while looking for the cup. It probably will not come back until some minutes, as it is new to the environment. After the robot knows your house, instead, you expect it to perform navigation tasks much faster, exploiting its previous knowledge of the environment to adapt to changes of objects, people, and furniture positioning. Embodied AI has recently gained a lot of attention from the research community, with amazing results in challenging tasks such as visual exploration [1], [2], [3] and navigation [4], [5], [6], [7]. However, in the current setting, the environment is completely unknown to the agent as a new episode begins. We believe that this choice is not supported by real-world experience, where information about the environment can be stored and reused for future tasks. As agents are likely to stay in the same place for long periods, such information may be outdated and inconsistent with the actual layout of the environment. Therefore, the agent also needs to discover those differences during navigation. In this paper, we introduce a new task for Embodied AI, which we name Spot the Difference. In the proposed setting, the agent must identify all the differences between an outdated map of the environment and its current state – a challenge that combines visual exploration using monocular images and embodied navigation with spatial reasoning. To succeed in this task, the agent needs to develop efficient exploration policies to focus on newly changed areas while exploiting priors about objects of the environment. We believe that this task could be useful to train agents that will need to deal with changing environments.

Recent work on Embodied AI has tackled the training of embodied agents capable of navigating and locating objects [8], [9], [10], [11]. One of the key factors for success in the field consists in building map representations in which knowledge about the environment can be stored while the agent proceeds [10], [4]. However, the dominant training and evaluation protocol involves an agent initialized from scratch that sees the environment for the first time [11]. Another line of work [12], [13], [14], [15], [16], instead, introduces a mapping phase of the environment to increase the performance on both exploration and downstream tasks. Unfortunately, if the environment changes over time, the agent needs to rebuild a full representation from scratch and cannot count on an efficient policy to update its internal representation of the environment. In this work, we simulate the natural evolution of an environment and design a specific policy to navigate in...
Due to the high cost of 3D acquisitions from the real world, the existing datasets of photorealistic 3D spaces [16], [17] do not contain different layouts for the same environment. In this paper, we create a reproducible set-up to generate alternative layouts for an environment. We semi-automatically build a dataset of 2D semantics occupancy maps in which the objects can be removed and rearranged while the area and the position of architectural elements do not change (Fig. 1). In the proposed setting, the agent is deployed in an interactive 3D environment and provided with a map from our produced dataset, which represents the information retained while performing tasks in a past state of the environment.

To train agents that can deal with changing environments efficiently, we develop a novel reward function and an approach for navigation aiming at finding relevant differences between the previous layout of the environment and the current one. Our method is based on the recent Active Neural SLAM paradigm proposed in [10] and [4]. Differently from previous proposals, though, it can read and update the given map to identify relevant differences in the environment in the form of their projections on the map. Our dataset and architecture can be employed with the Habitat simulator [18], a popular research platform for Embodied AI that renders photorealistic scenes and that enables seamless sim-to-real deployment of navigation agents [19], [20]. Experimental results show that our approach performs better than existing state-of-the-art architectures for exploration in our newly-proposed task. We also compare with different baselines and evaluate our agent in terms of percentage of area seen, percentage of discovered differences, and metric curves at varying exploration time budgets. The new dataset, together with our code and pretrained models, is available at this link.

II. RELATED WORK

Current research directions on Embodied AI for navigation agents can be categorized according to the quantity of knowledge about the environment provided to the agent prior to performing the task [11]. The first direction focuses on the scenario in which the agent is deployed in a completely new environment for which it has no prior knowledge [21], [10], [14]. Running exploration in parallel with a target-driven navigation task resulted in an effective approach to solve the latter (e.g., object-goal navigation [5] and point-goal navigation [4]). Other directions consider the case in which the agent can exploit pre-acquired information about the environment [22], [23] when performing a navigation task. Such pre-acquired information can be either partial [24], [25], [26] or complete [12], [8], [11]. A major limitation of such approaches is that the obtained map representation is assumed to conform perfectly with the environment where the downstream task will be performed.

In this work, we explore a fourth direction, in which the pre-acquired map provided to the agent is incomplete or incorrect due to changes occurred in the environment over time. Common strategies to deal with changing environments entail disregarding dynamic objects as landmarks when performing SLAM [27], [28] and applying local policies to avoid them when navigating [29]. An alternative strategy is learning to predict geometric changes based on experience, as done in [30], where the environment is represented as a traversability graph. The main limitation of this strategy is its computational intractability when considering dense metric maps of wide areas, as in our case.

III. PROPOSED SETTING

In the first part of this section, we introduce a new task for embodied agents, named Spot the Difference. We then describe the newly-proposed dataset that we create to enable this setting. Finally, we propose an architecture for embodied agents to tackle the defined task.

A. Spot the Difference: Task Definition

At the beginning of an episode, the agent is spawned in a 3D environment and is given a pre-built occupancy map $M$, representing its spatial knowledge of the environment, i.e. a previous state of the environment that is now obsolete:

$$M = (m_{ij}) \in [0, 1], \quad 0 \leq i, j < W,$$

where $m_{ij}$ represents the probability of finding an obstacle at coordinates $(i, j)$. The task entails exploring the current environment to recognize all the differences with respect to the state in which $M$ was computed, in the form of free and occupied space. To accomplish the task, the agent should build a correct occupancy map of the current environment starting from $M$, recognizing and focusing on parts that are likely to change (e.g., the middle of wide rooms rather than tight corridors).

For every episode of Spot the Difference, the agent is given a time budget of $T$ time-steps. At time $t = 0$, the agent holds the starting map representation $M$. At each time-step $t$, the map is updated depending on the current observation to obtain $M_t$. Whenever the agent discovers a new object or a new portion of free space, the internal representation of the map changes accordingly. The goal is to gather as much information as possible about changes in the environment by the end of the episode. To measure the agent performance, we compare the final map $M_T$ produced by the agent with the ground-truth occupancy map $M^*$. In this sense, the paradigm we adopt is the one of knowledge reuse starting from partial knowledge.

B. Dataset Creation

Semantic Occupancy Map. Given a 3D environment, we place the agent in a free navigable location with heading...
θ = 0° (facing eastward). We assume that the input consists of a depth image and a semantic image and that the camera intrinsics K are known. To build the Semantic Occupancy Map (SOM) of an environment, we project each semantic pixel of the acquired scene into a 2-dimensional top-down map: given a pixel with image coordinates \((i, j)\) and depth value \(d_{i,j}\), we first recover its coordinates \((x, y, z)\) with respect to the agent position. Then, we compute the corresponding \((u, v)\) pixel in map through an orthographic projection, using the information about the agent position and heading:

\[
\begin{bmatrix}
x \\
y \\
z \\
\end{bmatrix} = d_{i,j}K^{-1}\begin{bmatrix} i \\
j \\
1 \\
\end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} u \\
v \\
0 \\
1 \\
\end{bmatrix} = P_v\begin{bmatrix} x \\
y \\
z \\
1 \\
\end{bmatrix}. \tag{2}
\]

We perform the same operation after rotating the agent by \(\Delta_\theta = 30°\) until we perform a span from 0° to 180°. To cover the whole scene, we repeat this procedure placing the agent at a distance of 0.5m from the previous capture point, following the axis directions. The agent elevation is instead kept fixed. During this step, we average the results of subsequent observations of overlapping portions of space.

After the acquisition, we obtain a SOM with \(C\) channels, where each pixel corresponds to a 5em \(\times\) 5em portion of space in the 3D environment. For each channel \(c \in \{0, ..., C\}\), the map values represent the probability that the corresponding portion of space is occupied by an object of semantic class \(c\).

**Multiple Semantic Occupancy Maps for the Same Environment.** The SOMs obtained in the previous step can be seen as one possible layout for the corresponding 3D environments. In order to create a dataset with different states (i.e., different layouts) of the same environment, instead of manipulating the real-world 3D scenes (changing the furniture position, removing chairs, etc.), we propose to modify the SOM to create a set of plausible and different layouts for the environment.

First, we isolate the objects belonging to each semantic category by using an algorithm for connected component labeling \([31], [32], [33]\). Then, we sample a subset of objects to be deleted from the map and a subset of objects to be re-positioned in a different free location of the map. During sampling, we consider categories that have a high probability of being displaced or removed in the real world and ignore non-movable semantic categories such as fireplaces, columns, and stairs. After this step, we obtain a new SOM representing a possible alternative state for the environment, which could be very different from the one in which the 3D acquisition was taken. Sample manipulated maps can be found in Fig. 1.

**Dataset Details.** To generate alternative SOMs, we start from the Matterport 3D (MP3D) dataset of spaces \([16]\), which comprises 90 different building scans, and is enriched with dense semantic annotations. We consider each floor in the building and compute the SOM for that floor. For each map, we create 10 alternative versions of that same environment. In this step, we discard the floors that have few semantic objects (e.g., empty rooftops) or that are not fully navigable by the agent. As a result, we retain 249 floors belonging to 81 different buildings, thus generating a total of 2490 different semantic occupancy maps for these floors. Finally, we split the dataset into train, validation, and test subsets.

As an additional test bed, we also build a set of out-of-domain maps (13 floors from 5 spaces) taken from the Gibson dataset \([17]\), enriched with semantic annotations from \([34]\), and manipulated as done for the MP3D dataset. For each SOM, multiple episodes are generated by selecting different starting points. More information about our dataset can be found in Table I and in the supplementary material.

C. Agent Architecture

Our model for embodied navigation in changing environments comprises three major components: a mapper module, a pose estimator, and a navigation policy (which, in turn, consists of a global policy, a planner, and a local policy). An overview of the proposed architecture is shown in Fig. 2 and described below, while additional details can be found in the supplementary material. Although the data we provide is enriched with semantic labels, our agent does not make use of such information directly. This is in line with current state-of-the-art architectures for embodied exploration that we choose as competitors.

**Mapper.** The mapper module takes as inputs an RGB observation \(o^r_t\) and the corresponding depth image \(o^d_t\), representing
the first-person view of the agent at time-step \( t \), and outputs the agent-centric occupancy map \( v_t \) of a \( V \times V \) region in front of the camera. Each pixel in \( v_t \) corresponds to a 25mm×25mm portion of space and consists of two channels containing the probability of that cell being occupied and explored, respectively. As a first step, we encode \( o_t \) using the two first blocks of ResNet-18 pre-trained on ImageNet, followed by a three-layer CNN. We project the depth image \( o_t^d \) using the camera intrinsics [12] and obtain a preliminary map for the visible occupancy. We name the obtained feature representations \( o_t^f \) and \( o_t^d \), respectively. We then encode the two feature maps using a U-Net [35]:

\[
f_{\mu}(o_t^f, o_t^d) = \text{U-Net}_{\text{enc}}(o_t^f, o_t^d, \mu),
\]

and decode the \( 2 \times V \times V \) matrix of probabilities as:

\[
v_t = \sigma(\text{U-Net}_{\text{dec}}(f_{\mu}(o_t^f, o_t^d), \phi)),
\]

where \( \mu \) and \( \phi \) represent the learnable parameters in the U-Net encoder and decoder, respectively, and \( \sigma \) is the sigmoid activation function.

The computed agent-centric occupancy map \( v_t \) is then registered in the global occupancy map \( M_{t-1} \) coming from the previous time-step to obtain \( M_t \). To that end, we use a geometric transformation to project \( v_t \) in the global coordinate system, for which we need a triple \((x, y, \theta)\) corresponding to the agent position and heading in the environment. This triple is estimated by a specific component that tracks the agent displacements across the environment, as discussed in the following paragraph.

**Pose Estimator.** The agent can move across the environment using three actions: go forward 0.25m, turn left 10°, turn right 10°. Since each action may produce a different outcome because of physical interactions with the environment (e.g., bumping into a wall) or noise in the actuation system, the pose estimator is used to estimate the real displacement made at every time-step. We estimate the agent displacement \((\Delta x_t, \Delta y_t, \Delta \theta_t)\) at time-step \( t \) by using two consecutive RGB and depth observations, as well as the agent-centric occupancy maps \((v_{t-1}, v_t)\) computed by the mapper at \( t-1 \) and \( t \). The actual agent position \((x_t, y_t, \theta_t)\) is computed iteratively as:

\[
(x_t, y_t, \theta_t) = (x_{t-1}, y_{t-1}, \theta_{t-1}) + (\Delta x_t, \Delta y_t, \Delta \theta_t).
\]

We assume that the agent starting position is the triple \((x_0, y_0, \theta_0) = (0, 0, 0)\).

**Global Policy, Planner, and Local Policy.** The sampling of atomic actions for the exploration relies on a three-component hierarchical policy. The first component is the global policy, which samples a long-term global goal on the map. The global policy outputs a probability distribution over discretized locations of the global map. We sample the global goal from this distribution and then transform it in \((x, y)\) global coordinates. The second component is a planner module, which employs the A* algorithm to decode a local goal on the map. The local goal is an intermediate point, within 0.25m from the agent, along the trajectory towards the global goal. The last element of our navigation module is the local policy, which decodes the series of atomic actions taking the agent towards the local goal. In particular, the local policy is an RNN decoding the atomic action \( a_t \) to execute at every time-step. The reward \( r_t^{\text{local}} \) given to the local policy is proportional to the reduction in the Euclidean distance \( d \) between the agent position and the current local goal:

\[
r_t^{\text{local}} = d_t - d_{t-1}.
\]

Following the hierarchical structure, a global goal is sampled every \( N \) time-steps. A new local goal is computed if a new global goal is sampled, if the previous local goal is reached, or if the local goal location is known to be not traversable.

**Exploiting Past Knowledge for Efficient Navigation.** The global policy is trained using a two-term reward. The first term encourages exhaustive exploration and is proportional either to the increase of area-coverage [12] or to the increase of anticipated map accuracy as in [4]. Intuitively, the agent strives to maximize the portion of the seen area and thus maximizes the knowledge gathered during exploration. Moreover, since we consider a setting where a significant amount of knowledge is already available to the agent, we add a reward term to guide the agent towards meaningful points of the map. These correspond to the coordinates where major changes are likely to happen.

Given the occupancy map of the agent at time \( t \), \( M_t \), the true occupancy map for the same environment \( M^* \), and a time budget of \( T \) time-steps for exploration, we aim to minimize the following, for \( 0 < t < T \):

\[
D = \sum \mathbb{1}[M_t \neq M^*].
\]

In other words, we want to maximize the number of pixels in the online reconstructed map \( M_t \) that the agent correctly shifts from free to occupied (and vice-versa) during exploration. This leads to the reward term for difference discovery:

\[
r_{\text{diff}} = \sum \mathbb{1}[M_t = M^*] - \sum \mathbb{1}[M_{t-1} = M^*].
\]

The proposed reward term is designed to encourage navigation towards areas in the map that are more likely to contain meaningful differences (e.g., rooms containing more objects that can be displaced or removed from the scene). Additionally, an agent trained with this reward will tend to avoid difficult spots that are likely to produce a mismatch in terms of the predicted occupancy maps. This is because errors in the mapping phase would result in a negative reward.

To train our model, we combine a reward promoting exploration and the more specific reward on found differences to exploit semantic clues in the environment:

\[
r_t = \beta_1 r_{\text{exp}} + \beta_2 r_{\text{diff}}
\]

where \( r_{\text{exp}} \) is the reward term encouraging task-agnostic exploration (such as coverage-based or anticipation-based rewards, as described in the next section), and \( \beta_1 \) and \( \beta_2 \) are two coefficients weighing the importance of the two elements.
IV. EXPERIMENTS AND RESULTS

In this section, we detail our experimental setting and show experimental results for our new proposed task. Further analysis can be found in the supplementary material.

Evaluation Metrics. To evaluate the performance in *Spot the Difference*, we consider three main classes of metrics. First, we consider the percentage of navigable area in the environment seen by the agent during the episode (Seen\%). Then, we evaluate the percentage of elements that have been correctly detected as changed in the occupancy map (Acc.) and the pixel-wise Intersection over Union for the changed occupancy map elements (IoU). Besides, we evaluate the task as a two-class problem and compute the IoU score for objects that were added in place of free space (IoU+), and for objects that were deleted during the map creation (IoU−). In addition, to evaluate the performance independently from the exploration capability, we propose to compute the metrics only on the portion of space that the agent actually visited (mAcc, mIoU, mIoU+, and mIoU−).

Implementation Details. We conduct our experiment using Habitat [18], a popular platform for Embodied AI in photo-realistic indoor environments [17], [16]. The agent observations are 128 × 128 RGB-D images from the environment. The learning algorithm adopted for training is PPO [36]. The learning rate is $10^{-3}$ for the mapper and $2.5 \times 10^{-4}$ for the other modules. Every model is trained for $\approx 6.5$M frames using Adam optimizer [37]. A global goal is sampled every $N = 25$ time-steps. The local and global policies are updated, respectively, every $N$ and $20 \times N$ time-steps, and the mapper is updated every $4 \times N$ time-steps. The size of the local map is $V = 101$, while the global map size is set to $W = 2001$ for MP3D and to $W = 961$ for Gibson. The global policy action space size $G$ is 240. The reward coefficients $\{ \beta_1, \beta_2 \}$ are set to $\{10^{-2}, 10^{-1} \}$ when the exploration reward is based on coverage and anticipation reward, respectively. The length of each episode is fixed to $T = 1000$ time-steps.

Competitors and Baselines. We consider the following competitors and variants of the proposed method on two different setups: one where the agent position is predicted by the agent (as in Eq. [5]), and one where it has access to oracle coordinates: Difference Reward (DR): an exploration policy that maximizes the correctly predicted changes between $M$ and $M^*$. This corresponds to setting $\beta_1 = 0$ and $\beta_2 = 1$ in Eq. [9].

Coverage Reward (CR): an agent that explores the environment with an exploration policy that maximizes the covered area and builds the occupancy map as it goes, as in [4].

Anticipation Reward (AR): an agent that explores the environment with an exploration policy that maximizes the covered area and the correctly anticipated values in the occupancy map built as it goes, from [4]. Our proposed approach consists of an agent trained with the combination of the difference reward with the coverage reward (CR+DR) or with the anticipation reward (AR+DR).

Occupancy Anticipation (OccAnt): we also compare with the agent presented by Ramakrishnan et al. [4] using the available pre-trained models, referenced to as OccAnt. Note that OccAnt was trained on the Gibson dataset for the standard exploration task and without any prior map. Thus, it is not directly comparable with the other methods considered. We include it to gain insights into the performance of an off-the-shelf state-of-the-art agent on our task.

Results on MP3D dataset. As a first testbed, we evaluate the different agents on the MP3D *Spot the Difference* test set. We report the results for this experiment in Table II.

We observe that the agent combining a reward based on coverage and our reward based on the differences in the
starting map given to the agent as the episode begins, the results achieved by the CR agent, those of the proposed CR+DR agent, and the ground-truth map. The differences that the agents have correctly identified during the episode are highlighted in red. As it can be seen, the CR+DR agent can identify more differences than the CR counterpart, even in small environments (top row). As the size of the environments grows (bottom row), the performance gap increases and the CR+DR agent outperforms its competitor.

V. CONCLUSION

In this work, we proposed Spot the Difference: a new task for navigation agents in changing environments. In this novel setting, the agent has to find all variations that occurred in the environment with respect to an outdated occupancy map. Since current datasets of 3D spaces do not account for such variety, we collected a new dataset with different layouts for the same environment. We tested two state-of-the-art exploration agents on this task and proposed a novel reward term to encourage the discovery of meaningful information during exploration. The proposed agent outperforms the competitors and can identify changes in the environment more efficiently. We believe that the results presented in this paper motivate further research on this new proposed setting for Embodied AI.

ACKNOWLEDGMENT

This work has been supported by “Fondazione di Modena”, by the “European Training Network on PErsonalized Robotics as SErvice Oriented applications” (PERSEO) MSCA-ITN-2020 project (G.A. 955778), and by the “Artificial Intelligence for Cultural Heritage (AI4CH)” project, co-funded by the Italian Ministry of Foreign Affairs and International Cooperation.

### Table III

**Experimental results on Gibson validation set.**

| Estimated Localization | Seen [%] | Acc | IoUm | IoUr | IoUt | mAcc | mIoUc | mIoUt | mIoU |
|------------------------|----------|-----|------|------|------|------|-------|-------|------|
| OccAnt                 |          |     |      |      |      |      |       |       |      |
| DR                     |          |     |      |      |      |      |       |       |      |
| AR                     |          |     |      |      |      |      |       |       |      |
| CR                     |          |     |      |      |      |      |       |       |      |
| AR+DR                  | 84.3     | 58.7| 20.0 | 14.9 | 19.0 | 75.8 | 23.0  | 15.6  | 21.1 |
| CR+DR                  | 82.1     | 60.1| 24.0 | 19.0 | 23.1 | 78.5 | 27.8  | 19.9  | 25.9 |

Fig. 4. Qualitative results comparing the performances of the CR and CR+DR agents for different episodes.

environment (CR+DR) performs best on all the pixel-based metrics and places second in terms of percentage of seen area. It is worth noting that, even if the results in terms of the area seen are not as high as the ones obtained by the CR agent, the addition of our Difference Reward helps the agent to focus on more relevant parts, and thus, it can discover more substantial differences. Additionally, predictions are more accurate and more precise, as indicated by the 4.7% and 6.6% improvements in terms of Acc and IoU with respect to the CR competitor. Instead, a reward based on differences alone is not sufficient to promote good exploration. In fact, although the DR agent outperforms the CR and AR agents on some metrics, our reward alone does not provide as much improvement as when combined with rewards encouraging exploration (as for CR+DR and AR+DR).

Even in the oracle localization setup, the CR+DR agent achieves the best results. Interestingly, the gap with the CR agent decreases to 1.1% and 0.1% in terms of Acc and IoU, respectively. This is because our CR+DR agent learns to sample trajectories that can be performed more efficiently and without accumulating a high positioning error. For this reason, the performance boost given by the oracle localization is lower. For both setups, our CR+DR agent outperforms the state-of-the-art OccAnt agent for exploration on all the metrics.

Finally, in Fig. 3, we plot different values of Acc and IoU over different time-steps during the episodes. This way, we can evaluate the whole exploration trend, and not only its final point. We can observe that the proposed models incorporating the difference reward outperform the competitors. In particular, the CR+DR agent scores first by a significant margin. The CR agent is the second-best in terms of Acc and IoU, respectively. This is because our CR+DR agent learns to sample trajectories that can be performed more efficiently and without accumulating a high positioning error. For this reason, the performance boost given by the oracle localization is lower. For both setups, our CR+DR agent outperforms the state-of-the-art OccAnt agent for exploration on all the metrics.

**Results on Gibson dataset.** The environments from the Gibson dataset [17] are generally smaller than those in MP3D, and thus, they can be explored more easily and exhaustively. We report the results for this experiment in Table III. Also in this experiment, the CR+DR agent performs best on all the metrics but the percentage of the area seen. Although CR+DR explores 3.8% of the environment less than the CR agent, it still overcomes the competitor by 2.5% and 7.7% in terms of Acc and IoU. The AR+DR agent is the second-best in terms of Acc. The OccAnt agent, instead, is competitive in terms of area seen but achieves low Acc. and IoU metrics.

**Qualitative Results.** In Fig. 4, we report some qualitative results. Starting from the left-most column, we present the
