Visual Rearrangement in Embodied AI with 3D Gaussian Splatting and Dense Feature Matching

Arjun P S
IIT ISM Dhanbad
papers.arjun@gmail.com

Andrew Melnik
Bremen University, Germany

Gora Chand Nandi
IIIT Allahabad

Abstract: Recent advances in volumetric scene representation like 3D Gaussian Splatting, offer fast rendering of high quality and photo-realistic novel views. In this work, we present a novel framework that leverages on 3D Gaussian Splatting as a 3D scene representation for visual rearrangement task in Embodied AI. Our approach enables the agent to have consistent views of the current and the goal setting of the rearrangement task. To compare these images, we propose to use a dense feature matching method with visual features extracted from a task-agnostic and self-supervised foundation model like DINOv2. We validate our approach on the AI2-THOR rearrangement challenge benchmark and demonstrate improvements over the current state-of-the-art methods.
Visualization: `https://youtu.be/8u3c1lbVFeo`

Keywords: Gaussian Splatting, Visual Rearrangement, Embodied AI

1 Introduction

Rearrangement challenge [1] is a pivotal benchmark in developing Embodied AI agents that can interact with the physical world. It involves navigating through complex scenes, exploring and recognizing the current state of the world, reasoning about the changes in the world from a goal state and manipulating objects in the regions of change to bring it to the goal state where the goal specification can be represented in terms of a geometric transformation, image, language, experience or a predicate. This work focuses on the experience goal setting, where the agent is immersed in the environment at its goal state, letting the agent build a representation of the world. The agent is then initialized in a shuffled state of the same environment and is tasked to rearrange it to the goal state.

Our work uses 3D Gaussian Splatting [2] as a 3D scene representation for the rearrangement task. 3D Gaussian splatting based volumetric representation enables fast and real-time rendering of the scene by optimizing a set of 3D Gaussian primitives and is capable of synthesizing high quality and photorealistic novel views. By training a Gaussian Splat of the goal setting, the agent can freely explore the scene and render the corresponding viewpoint from the splat using a virtual camera, thus enabling the agent to have consistent view of the current and goal setting. Prior works have focused on modelling the scene with a variety of methods: scene graphs where nodes are objects and the relation between objects are represented with edges [3, 4], 3D voxelized maps [5], 2D semantic maps [3], pointcloud based representation [6] and images [7].

Consistent images from the current observation frame and rendered view enable a direct comparison. Even though the images are similar in their visual aspects, they are different at the pixel level. Image from the trained Gaussian Splat is rendered by rasterizing 3D Gaussians to a 2D plane, which might create artifacts in the rendered image. Considering this, We implement a patchwise dense feature matching method with patch-level features extracted from DINOv2 [8] to compare the two images. Changes in the scene can be recognized by matching the features of corresponding patches across the current and the rendered frame.

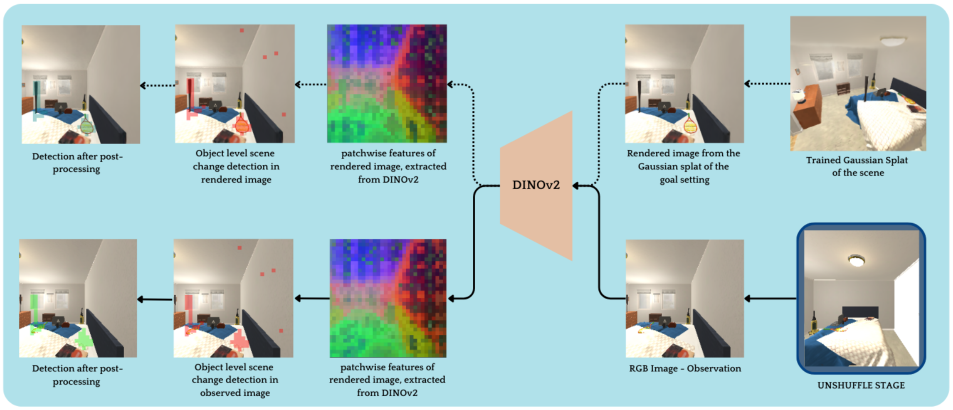
Detection after post-processing
Object level scene change detection in rendered image
patchwise features of rendered image, extracted from DINOv2
DINOv2
Rendered image from the Gaussian splat of the goal setting
Trained Gaussian Splat of the scene
Detection after post-processing
Object level scene change detection in observed image
patchwise features of rendered image, extracted from DINOv2
RGB Image - Observation
UNSHUFFLE STAGE

Figure 1: Overview of our approach. Images, observed by the agent and rendered from the Gaussian Splat(trained on the goal setting) with a virtual camera at the same location as that of the agent are compared with a patchwise feature matching method. The resulting detections are stored as an object node. The patchwise feature visualization above is generated by taking the PCA (principal component analysis) of combined features in image of the current and goal setting.

2 Technical Approach

Our method builds a 3D Gaussian Splatting based 3D scene representation for experience goal rearrangement task. During the experience collection phase, the agent collects RGB image, camera trajectory and pointcloud information. At the end of this phase, the agent trains a 3D Gaussian Splat to represent the scene. The agent is then immersed in the same environment with shuffled object configurations. As the agent explores the shuffled scene, the agent uses a virtual camera in the trained Gaussian Splat to render images of the goal configuration. By using patch-wise dense feature matching with DINOv2 , the agent recognizes the regions in the current observed image, which differ from the goal setting and vice-versa. These regions and their spatial and semantic information are stored separately as objects in the current and goal setting. At the end of explorations, the agent uses a category agnostic semantic matching method, to find objects in the current scene configuration that are similar to objects in the goal configuration. The agent uses these matched object pairs to perform the rearrangement action.

2.1 Exploration and Data collection

Intelligent exploration is crucial in collecting high quality data for training the Gaussian Splat. The agent must ensure that the scene is fully explored and well observed to avoid any artifacts. To ensure this, the agent incrementally builds a 2D map of the scene, we leverage the 2D map module from [3] to keep track of obstacles in the scene. The exploration strategy is designed in such a way that the agent covers the entire scene. This is done by randomly sampling a point on the unexplored traversable regions of the map at that timestep, using a fast-marching method [9] to navigate to this point. This is continued until all the traversible regions have been explored. The agent collects RGB image, pose information and pointcloud data for every frame and at the end of exploration, the pointcloud is downsampled to a lower resolution by averaging points within a specified voxel grid size. This is done to limit the amount of Gaussians initialized during training, which reduces the overall computational overhead.

2.2 Gaussian Splatting

The agent builds a Gaussian Splat based volumetric representation [2] of the scene, to perform the rearrangement task. This representation is constructed by initializing a set of Gaussians in the scene and optimizing their parameters. Each Gaussian g is parameterized by its mean $\mu_g \in R^3$, 3D covariance Σ_g that is parameterized by a scaling vector $s_g \in R^3$ and rotation matrix R_g, opacity

$\alpha_g \in [0,1]$ and RGB color C which is represented by a set of spherical harmonic coefficients. For a point $x \in R^3$ in 3D space, the 3D Gaussian equation and the parameterized covariance :

$$G_g(x) = e^{-\frac{1}{2}(x-\mu_g)^T \Sigma_g^{-1}(x-\mu_g)} \tag{1}$$

$$\Sigma = RSS^T R^T \tag{2}$$

2.3 Patch-Wise Dense Feature Matching

We utilize patch-level features extracted from the DINOv2 [8] model to recognize changes in the image. DINOv2 follows a vision transformer architecture where an image input returns a class token, patch tokens with patch size 14 and optionally 4 register tokens. Images from the current I^C and goal I^G setting are passed through the DINOv2 model to get their corresponding patch-wise features f^C and f^G. We compute the cosine similarity S_{ij} between extracted features f^C_{ij} and f^G_{ij} of corresponding patch (i,j) in the current and goal configuration, to determine if those patches are similar or not. We group adjacent similar patches, that are dissimilar across the current and rendered image to form objects.

2.4 Post-Filtering Techniques

The detections retrieved from the patchwise feature matching method can often be noisy. This is generally due to artifacts in the trained Gaussian Splat. It has been observed that these artifacts generally occur on plain or reflecting surfaces like walls or mirrors. Since the agent has access to all the object names in the RoomR dataset, to address the noisy detections, the agent computes the similarity (cosine similarity of CLIP feature vectors of the region and object names) of the detected region with the list of objects conditioned with the words ”wall” and ”mirror”. If the computed similarity is highest for the conditioned terms, it will likely be a false detection. We also reject all the one-patch sized detections, as most noisy detections are of this size. By doing so, it is highly unlikely to miss any true detection of an object, as the navigation algorithm is designed to move closer to all the objects in the scene, thus providing detections that are more than one-patch in size.

2.5 Object Nodes

For every object level detection, the agent stores an object node for the shuffled and goal setting separately. This object node consist of the corresponding image I, mask m, semantic information: CLIP [10] embedding g of an image cropped along the objects mask and spatial information: center coordinate p and pointcloud P in world space, of the object. Since we use patchwise features, the masks for objects in the scene are not accurate. To address this, we use segment anything model [11] (SAM) to find the accurate mask, by prompting the image with a bounding box prompt, where the bounding box is along the mask obtained from grouped DINOv2 patches. For each incoming object node, the agent calculates a similarity Φ^{node} index based on its visual Φ^{vis} and spatial similarity Φ^{sp} to the nodes stored in memory, to either register it as a new detection or merge it with an existing one. For an incoming object node O and a node in the memory O_i

$$\Phi_i^{node}(O, O_i) = \delta \Phi_i^{vis}(O, O_i) + (1-\delta)\Phi_i^{sp}(O, O_i) \tag{3}$$

where δ is a weighting factor.

$$\Phi_i^{vis}(O, O_i) = \frac{g \cdot g_i}{\|g\| \|g_i\|} \tag{4}$$

$$\Phi_i^{sp}(O, O_i) = nnratio(P, P_i) \tag{5}$$

where $nnratio$ [12] is the proportion of points in the pointcloud P, that have nearest neighbors in pointcloud P_i of the i-th object O_i.

Methods	% Fixed Strict↑	% Misplaced↓	% Energy Remaining↓	Success rate↑
Ours	**0.3025**	**0.6974**	**0.7112**	0.0384
TIDEE [3] + open-everything	0.2894	0.7347	0.7153	**0.117**
CAVR [6]	0.2425	0.7968	0.8007	0.0890
MaSS [5]	0.1656	1.0187	1.0165	0.0470

Table 1: Comparison of results obtained with SOTA baselines.

3 Experimental Evaluation

We use the AI2-THOR rearrangement challenge as a benchmark [7] for evaluating our method. This challenge is based on the Room Rearrangement dataset (RoomR), built on AI2-THOR [13] virtual environment. The RoomR dataset consist of 6000 unique rearrangement tasks with 120 rooms that include kitchen, living room, bathroom, and bedroom, and more than 70 unique object categories. This challenge has two tracks: 1-Phase and 2-Phase task. We test our approach on 2-Phase task of the AI2-THOR rearrangement challenge, which comprises of two stages: Walkthrough stage, where the agent is immersed in the goal setting and Unshuffle stage, where the agent is initialized in the same scene with shuffled object states and the agent is tasked to rearrange them to their initial states. For each episode there can be 1-5 objects whose state has been changed in the unshuffle task. State changes can occur either through changes in the object's position or through variations in its degree of openness. The degree openness quantifies, how far an openable object like cabinet is opened or closed. To complete a rearrangement task, the agent will have to replace all the objects whose pose has been altered and change the degree of openness of all the objects to what it was during the walkthrough stage.

Evaluation Metrics. Success rate is a binary metric that indicates whether the agent has successfully completed the task or not.% Fixed Strict is the ratio of number of objects that were rearranged successfully to the number of objects in the shuffled configuration at the start of the episode.% Misplaced is the ratio of number of objects misplaced at the end of an episode in unshuffle task to the number of objects misplaced at the start of that task. % Energy Remaining, is used to provide the agent with partial credit, for partially completing a task. This metric is computed as the ratio between the amount of energy at the end of the rearrangement task to that of the start. The energy function is a distance metric which increases from zero as the current object pose moves away from the goal pose.

4 Results and Discussion

We compare the performance of our method with other state-of-the-art (SOTA) methods in Table 1. Our approach improves upon the current SOTA on % Fixed Strict, % Misplaced and % Energy Remaining. Improvement to fixed strict suggests that our method was able to successfully rearrange more objects in the scene, compared to other SOTA methods. With a lower misplaced percentage than other methods, The agent was able to undergo rearrangement task with minimal disruption to objects that were not shuffled in the unshuffle phase. Lower value of % energy remaining suggests that our agent rearranged more objects close to its goal location, without entirely completing the task.

5 Conclusion

We present a novel approach to solve experience goal based rearrangement task, that uses 3D Gaussian Splat as a 3D scene representation for robotics. With a 3D Gaussian Splat of the goal setting, the agent can freely explore the shuffled scene with the guarantee that it can render a consistent image of the environment in the goal state. We test our approach on the AI2-THOR rearrangement challenge benchmark and show that our approach provides improvement over the current SOTA methods. We demonstrate that our work opens up innovative and exciting avenues in using 3D Gaussian Splatting as a world model [14] for Embodied AI [15].

Acknowledgments

This work was supported by DAAD PPP-India program under project number 57622912.

References

[1] D. Batra, A. X. Chang, S. Chernova, A. J. Davison, J. Deng, V. Koltun, S. Levine, J. Malik, I. Mordatch, R. Mottaghi, M. Savva, and H. Su. Rearrangement: A challenge for embodied ai, 2020. URL `https://arxiv.org/abs/2011.01975`.

[2] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. URL `https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/`.

[3] G. Sarch, Z. Fang, A. W. Harley, P. Schydlo, M. J. Tarr, S. Gupta, and K. Fragkiadaki. Tidee: Tidying up novel rooms using visuo-semantic commonsense priors, 2022. URL `https://arxiv.org/abs/2207.10761`.

[4] S. Y. Gadre, K. Ehsani, S. Song, and R. Mottaghi. Continuous scene representations for embodied ai, 2022. URL `https://arxiv.org/abs/2203.17251`.

[5] B. Trabucco, G. Sigurdsson, R. Piramuthu, G. S. Sukhatme, and R. Salakhutdinov. A simple approach for visual rearrangement: 3d mapping and semantic search, 2022. URL `https://arxiv.org/abs/2206.13396`.

[6] Y. Liu, X. Song, W. Li, X. Wang, and S. Jiang. A category agnostic model for visual rearrangment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16457–16466, June 2024.

[7] L. Weihs, M. Deitke, A. Kembhavi, and R. Mottaghi. Visual room rearrangement, 2021. URL `https://arxiv.org/abs/2103.16544`.

[8] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. URL `https://arxiv.org/abs/2304.07193`.

[9] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996. doi:10.1073/pnas.93.4.1591. URL `https://www.pnas.org/doi/abs/10.1073/pnas.93.4.1591`.

[10] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision, 2021. URL `https://arxiv.org/abs/2103.00020`.

[11] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[12] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa, C. Gan, C. M. de Melo, J. B. Tenenbaum, A. Torralba, F. Shkurti, and L. Paull. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning, 2023. URL `https://arxiv.org/abs/2309.16650`.

[13] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, M. Deitke, K. Ehsani, D. Gordon, Y. Zhu, A. Kembhavi, A. Gupta, and A. Farhadi. Ai2-thor: An interactive 3d environment for visual ai, 2022. URL `https://arxiv.org/abs/1712.05474`.

[14] A. Melnik, F. Schüler, C. A. Rothkopf, and P. König. The world as an external memory: The price of saccades in a sensorimotor task. *Frontiers in behavioral neuroscience*, 12:253, 2018.

[15] P. König, A. Melnik, C. Goeke, A. L. Gert, S. U. König, and T. C. Kietzmann. Embodied cognition. In *2018 6th International Conference on Brain-Computer Interface (BCI)*, pages 1–4. IEEE, 2018.