Hindsight Experience Replay in the Visual Domain With Novel View Synthesis Networks

Eric Zhu[1], Mara Levy[1], Abhinav Shrivastava[1]

***Abstract*— Learning from failed trajectories is a long standing problem in reinforcement learning and robotics. Reinforcement learning policies that fail to achieve a goal often do not learn any meaningful information about how to achieve the goal from these failed trajectories. This phenomenon results in inefficient learning because the majority of the data training data are failed demonstrations. To solve this problem, hindsight experience replay has been proposed. While hindsight experience replay has been shown to be effective for vector-based state representations of the environment, there has not been any work on applying hindsight experience replay to the image domain. In this paper, we bring the original hindsight experience replay algorithm into the image domain by using NeRF-based editing to effectively change the position of goal objects in an image. Overall, we show that applying hindsight experience replay increases the success rate of the reinforcement learning policies by between 40% and 80% on 4 different tasks.**

I. Introduction

Humans have an innate ability to understand why the actions they took failed at achieving a task. A human can often list the reasons that the approach failed and imagine a scenario in which they succeeded. Even little children exhibit this ability to explain why they could not do basic tasks such as holding a toy correctly or why a object slipped from their hand. This is in stark contrast to existing reinforcement learning models which are often incapable of explaining why their actions fails. This lack of "hindsight" leads to significant overhead in training a reinforcement learning model because the policy does not learn new information from previous failed attempts to solve the task.

In typical reinforcement learning, for most of training, the memory buffer is filled with many failed executions compared to a relatively small number of (if any) successful ones. Such a scenario is suboptimal because there is not much successful data to learn from, leading to significantly longer training time and environment steps required to create a successful policy. Such a problem is exacerbated in sparse reward settings where a policy gets a flat reward signal until it has by chance successfully completed the task, if ever.

To solve this issue, hindsight experience replay (HER) has been proposed. Hindsight experience reply is a method that takes a trajectory that failed to achieve its goal and creates a new trajectory that did successful reach its goal. By doing so, HER effectively creates a new synthetic successfully trajectory from failed trajectories.

HER has been proven to be effective in a multitude of cases. [1] showed that the original hindsight experience algorithm was capable of training a policy to (1) reach a point in mid-air, (2) push a puck to a correct position, (3) slide a puck to a target position, and (4) pick up and move a block to a new position. However, hindsight experience replay has only been applied to vector representations of states, where policy has access to gripper and object positions and a goal of the form (x, y, z). Such representations are highly limited because it requires the ground truth position of all the objects in a scene to execute a task, which is unfeasible at test time.

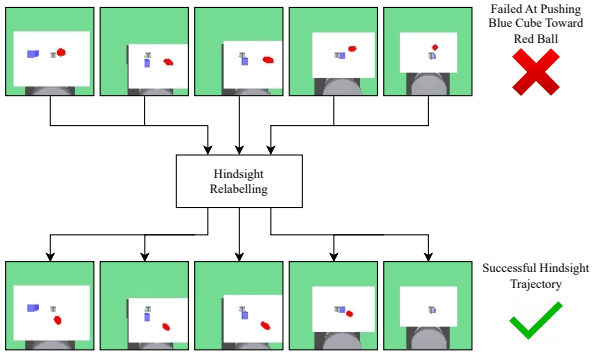

Fig. 1. An overview of how our method is able to change the positions of objects around in a scene. We relabel a trajectory taken by the policy in order to get a new successful trajectory even if the original trajectory was a failure.

In this paper, we propose a pipeline to apply HER to the image domain. We leverage novel view-synthesis methods to edit images to convert failed trajectories into synthetic trajectories that succeeded. Our contributions are as follows:

- We learn a NeRF or other novel view synthesis model on a target object.
- We provide a simple but effective pipeline to allow agents to imagine objects in different places.
- We use this pipeline to convert failed trajectories into successful trajectories by changing the location of objects in a scene in the same way as the original hindsight experience replay paper.
- We demonstrate our method's effectiveness on multiple environments from drone navigation to robot arm manipulation.

II. Related Works

A. Novel View Synthesis

Novel view synthesis methods are models that are capable of rendering a scene or object from a point of view not present in the dataset. Over the past few years, novel view synthesis methods have improved immensely in both speed

[1]University of Maryland, College Park

and photorealism. The first novel view synthesis method was the neural radiance field (NeRF)[2], which estimates the rgba value of each position in space and then performs ray tracing to render images. Many modifications have been proposed to NeRF. These include methods that significantly improving the speed of NeRF rendering, proposed by [3**?** –6]. Others methods have explored diffusion-guided NeRF training from a single viewpoint [7], hardening NeRF models to noise [8], creating NeRF models that capable of reflections [9], and dynamic NeRf models for capturing a 3D scene where objects are moving.[10].

Instead of using a mlp for predicting rgba values, other novel view synthesis methods use other trainable structures. [11] replaces the traditional mlp with a trainable octree, which provides both faster training and inference time than a NeRF model. Similarly , [12] uses a 3D voxel grid structure with trainable cubes to render the rgba values of an image.

Recently, gaussian splatting [13] has gained popoularity as an alternative to NeRF models for its higher image quality and faster render time. Gaussian splatting has been used for dynamic scenes [14], 3D segmentation [15, 16], and scene editing [16].

B. Novel View Synthesis in Robotics

Novel View Synthesis has gained a following in robotics due to their strong 3D consistency and photorealism. To bridge the sim-to-real gap, both NeRFs and Gaussian Splats have been proposed to accurately render a real object's texture on top of a simulator to thereby create photorealistic images in simulation [17–19]. Other methods leverage NeRF's ability to estimate the transparency of an object in order to create an accurate and complete depth map for grasp pose estimation [20, 21]. For data augentation, NeRF models have been used to create synthetic data for a policy to train on [22, 23]. NeRF models have been effective in estimating the 6-dof position/orientation of a target object, as shown in [24–26]. Linguistically-augmented novel view synthesis methods for both manipulation [27, 28] and navigation [29] have also been proposed. Finally, novel view synthesis has been combined with slam algorithms for efficient and photorealistic reconstruction [30, 31].

C. Replay Buffers and Hindsight Experience

There have been many variations of the original hindsight experience replay algorithm [1] that either modify the trajectories or change the priority of transitions sampled from the replay buffer. In [32], the authors propose to change the priority of the transitions in the replay buffers by sampling for values with a higher TD error. Other approaches combine hindsight with model-based reinforcement learning by using the model to generate a new trajectory and rewrite the goal of said trajectory [33]. The authors of [34] take an unoptimal trajectories and perform a graph search to find a more efficient trajectory. Other approaches leverage diffusion models to upsample data in a replay buffer [35, 36]. Realizing that trajectories often result in survivor bias where trajectories that the policy runs more often have a larger density of successful synthetic trajectories, [37] unbias the sampling of a hindsight replay buffer. Other approaches change the task of an executed trajectory if it is not the target task [38].

III. Preliminaries

Reinforcement learning. In the reinforcement learning paradigm, a policy continuously interacts with the environment in order to gain data to train on. Each time the policy interacts with the environment, it receives a reward value, and the policy's goal is to optimize the total reward that it receives.

Formally, we model the reinforcement learning policy as a markov decision process defined by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, p_0, \gamma)$. $\mathcal{S}$ is the set of states of the environment, $\mathcal{A}$ is the set of actions, $\mathcal{P}$ is a probabilistic transition function from a state-action tuple to a new state, $\mathcal{R}$ is the reward function mapping a state-action tuple to a reward, and γ is the decay factor. A trajectory is defined as $\tau = \left\{(s_k, a_k, r_k, s_{k+1})\right\}_{k=0}^{n}$. The goal of the policy is to optimize the sum of decayed rewards denoted by $\sum_{i=0}^{\infty} \gamma^i R(s_i, a_i)$.

In goal-conditioned reinforcement learning, we add a goal element $g_k \in \mathcal{S}$ for a trajectory of the form $\tau = \left\{(s_k, a_k, r_k, s_{k+1}, g_k)\right\}_{k=1}^{n}$. A trajectory is deemed successful if $s_n = g_n$. Moreover, g_k typically stays the same for the entire trajectory.

Hindsight Experience Replay. Hindsight experience replay is a way of modifying trajectories in goal conditioned reinforcement learning. For a trajectory $\tau = \left\{(s_k, a_k, r_k, s_{k+1}, g_k)\right\}_{k=1}^{n}$, we change all the goal values g_k to be the final end state s_n. Hindsight experience replay also assumes enough knowledge of the reward function to rewrite the reward to be accurate with the new goal. The result is the trajectory

$$\tau' = \left\{(s_k, a_k, R(s_k, a_k, g_k), s_{k+1}, s_n)\right\}_{k=1}^{n}$$

Optionally, hindsight experience replay may modify only subtrajectories rather than an entire trajectory. These new successful trajectories are then added to the replay buffer of the reinforcement learning policy.

Novel View Synthesis. The basic assumption of novel view synthesis access to a dataset of image-camera position pairs $D = \{(I, T)\}_{k=1}^{K}$. This dataset can be of a scene (scene-level) or an object (object-level). A model learns from this dataset and renders an image of the scene/object at a new camera pose. Some prominent novel view synthesis models include gaussian splats, neural radiance fields, and plenoxels. We choose to use plenoxels due to its ease of use, real-time render speed, and robustness to many different viewpoints, although any of the aforementioned methods would work with our pipeline.

IV. Our Approach

Our key observation is that the goal in many environments is an object that a robot either navigates towards or interacts with. In this way, an environment with images is implicitly a goal conditioned reinforcement learning problem where the object is the goal. Therefore, we propose a pipeline where we

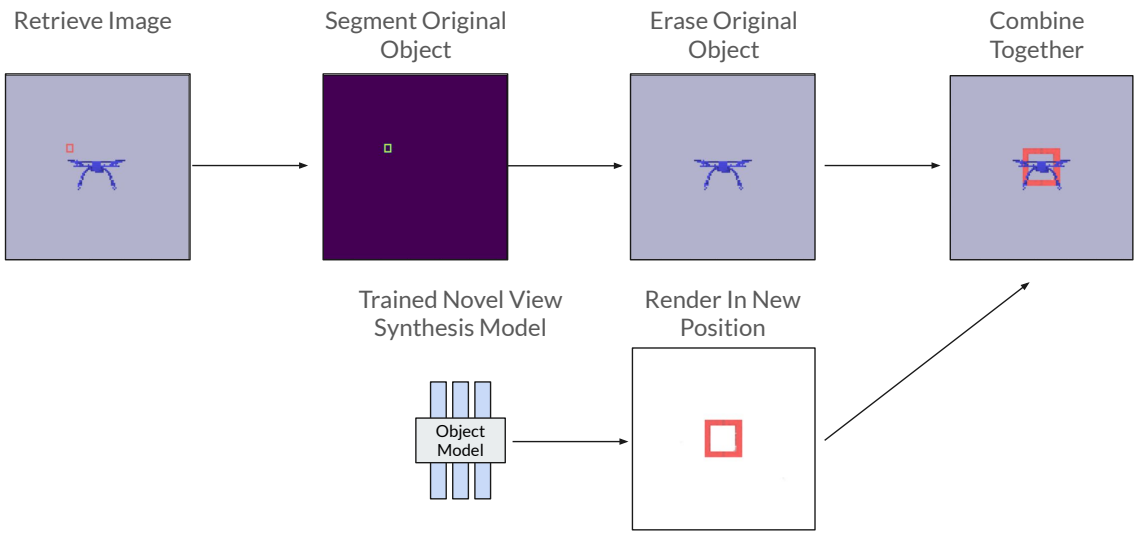

Fig. 2. Our method for rewriting the position of an object in a trajectory. We show the accuracy of our method on the "Navigate Drone" task. By leveraging a NeRF or similar model to synthesize the object in a new position. We observe that our method is extremely photorealistic, allowing for synthetic trajectories to be indistinguishable from true trajectories. Images are from our "Navigate Drone" environment.

use the final end state of the robot to rewrite the a trajectory in the same fashion as the original hindsight experience replay. We breakdown our method into 5 steps as seen in fig 2 and listed below:

- We learn an object-level model that is capable of synthesizing new viewpoints of the object.
- We segment the object of interaction in each image of the trajectory and erase the object.
- We find a new goal/object position using the end position of the robot.
- We calculate the matrix of the camera(s) relative to the new position of the object.
- We render the object at a new position and blend the outputs of the novel view synthesis model into the new trajectory.

A. Creating an Object Level Model

Creating an NeRF model for an object is a widely used practice in robotics. This can be done by using a robot gripper arm to scan an object or a drone camera flying around one. For our project, we randomly selected about 100 pictures from various viewpoints as training data for our novel-view-synthesis model. While there are many different novel-view-synthesis models, we chose to use plenoxel because of its ability to render at high speed with good resolution. Since our policy is learning online, having a model that can render in real time is crucial. Moreover, plenoxel also synthesizes accurate depth maps which can be useful for modifying rgbd and depth map based policies.

B. Segment and Removing The Original Object

Like previous works, we assume access to the pixel level segmentation of the original object in a trajectory. If such segmentation is not available, there is a rich set of image and video segmentors that are capable of segmenting an object in a dataset of images. For the extent of this work, we will use Pybullet and Mujoco's built-in segmentation masks that come with their respective renderers.

After the mask is obtained, we erase the original object. There are multiple off-the-shelf inpainters capable of erasing an object that previous methods have used. For the sake of simplicity, we use a reference image of the scene and then replace the pixels in the mask with the background image. This process can also be done for depth masks.

C. Calculate Camera To Object Matrix

For a trajectory / subtrajectory that we want to image edit, we calculate the new desired position of the object. Let the new position and orientation of the object relative to the world be described by the transformation matrix $T_{\text{object-to-world}}$. Thus,

$$T_{\text{object-to-camera}} = T^{-1}_{\text{camera-to-world}} \cdot T_{\text{object-to-world}}$$

Note that this formulation is valid for both 3rd person cameras and dynamic first person cameras. We show the effectiveness of our method in both scenarios in our Results section.

D. Blend Model Renderings With Original Image

We pass $T_{\text{object-to-camera}}$ into the novel view synthesis model to produce a rendering of the object in the new position, denoted I_{rendered}. We also obtain an object mask for this newly rendered object, denoted M_{rendered}. We blend the rendered image the image with the object erased through the following method:

$$I_{\text{final}} = I_{\text{rendered}} \odot M_{\text{rendered}} + I_{\text{object\_erased}} \odot \left(1 - M_{\text{rendered}}\right)$$

If the observation space include the depth images, we can optionally blend the rendered depth image with the original depth image in the same way.

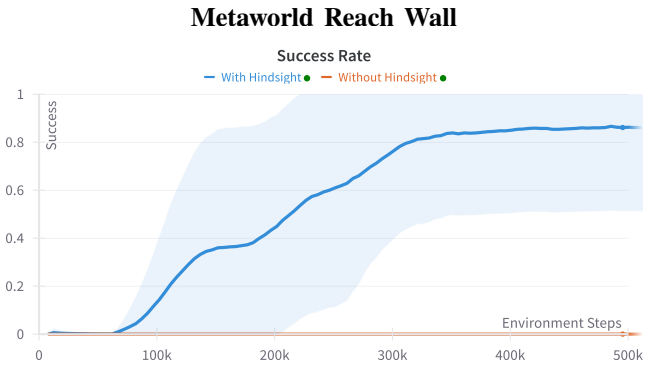

Fig. 3. The success rate for the "Metaworld reach wall" task where a robot arm has to reach over a wall and onto a ball that is floating in mid-air.

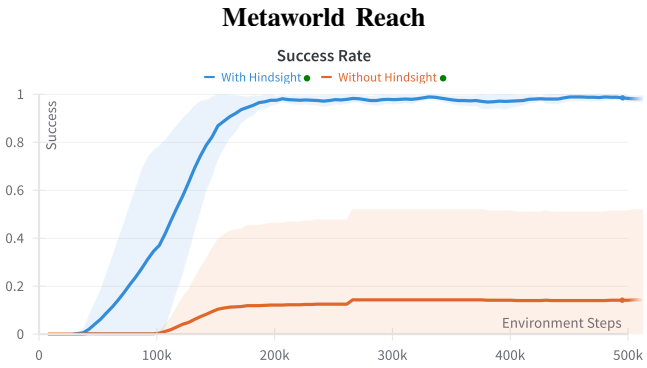

Fig. 4. The success rate for the "Metaworld Reach" task where a robot arm has to reach towards a ball in the air.

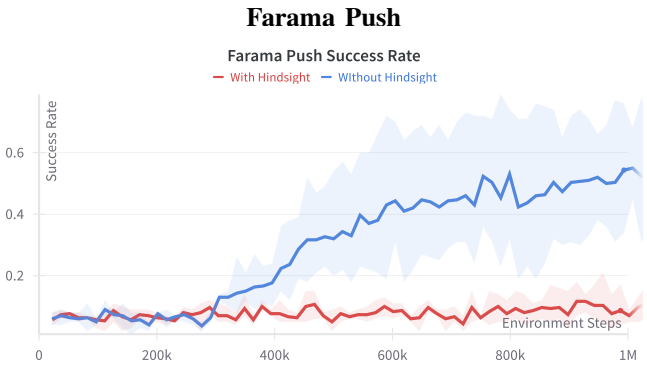

Fig. 5. The success rate for the "Farama Push" task where a robot arm to push a puck to a target location.

E. Adding New Trajectories

We generally found that turning about 30% of trajectories into relabeled trajectories was the most effective in achieving good results. We sample a new goal randomly from each position of the trajectory. Additionally, we only sample goals in the relevant goal space of the gym environment (ie if the robot goes out of bounds where a goal can be, we do not sample from those positions for the new goal of the trajectory).

V. Experiments

A. Environments

We test our method on 4 different environments from Metaworld [39], Farama [40], and Pybullet [41], as listed below:

1. Drone Navigation
2. Metaworld Reach
3. Metaworld Reach Wall
4. Farama Push

In the "Drone Navigation" environment, a policy is meant to navigate an aerial drone to go through an orange square hoop. The camera is located behind the drone and follows

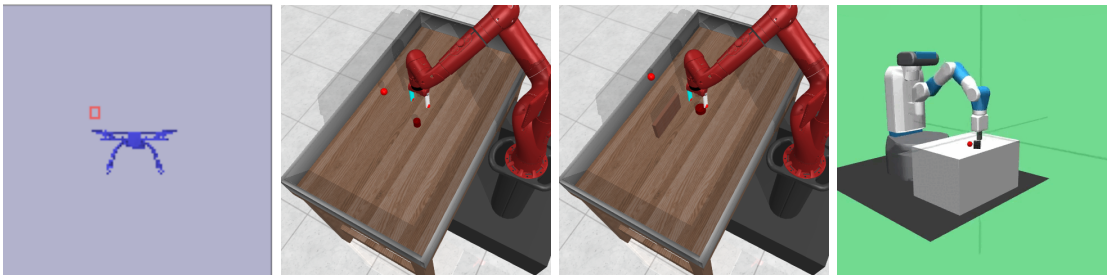

Fig. 6. Our 4 environments. From left to right, the first one is the Drone Navigation task, the second one is the Metaworld Reach task, the third one is Metaworld Reach Wall task, the final one is Farama Push task. The difference between Metaworld Reach and Metaworld Reach Wall is the presence of a brick wall in front of the robot.

the drone as it navigates through the hoop. We create this environment by ourselves using Pybullet [41]. The action space was 3 values for the x,y,z velocity of the drone, and the observation space was a rgb image. We used a sparse reward for this environment and an episode ended when the drone either flew through the hoop for or the hoop left the field of view of the camera. Refer to fig. 6 for more details.

In the "Metaworld Reach" and "Metaworld Reach Wall" tasks, a Sawyer robot arm attempts to reach a static ball that is positioned above the table that the robot arm is next to. We change the reward to be a sparse signal with a tolerance value of 0.025. In order to be more visible, we change the ball to be about twice the size of the original ball and switched the color from red to green to prevent it from blending in with the red robot arm. To clarify, these modifications do not change the dynamics of the environment, only the visuals. Additionally, we change the 3rd person camera to be parallel to the side of the table for a better view of the situation. For both environments, we keep the original 3-dof action space from Metaworld but change the observation space to be a rgbd image. Refer to fig. 6 for more details.

In the "Farama Push" task, we use the original push task from the original hindsight experience replay paper. Namely, a Farama robot arm is tasked with pushing a cube on a table into a red "target" circle. Again, we change the size of the target object to be more visible to the camera. To clarify, this does not change the dynamics of the environment, only the visuals. We also we used first person camera that is attached to the gripper and changed the position so that it could see the entire table. We use the original sparse reward option that comes from the Farama environment. Refer to fig. 6 for more details.

B. Results

As seen in fig 3-5, adding our visual HER method to the standard soft-actor critic policy [42] results in significantly faster convergence of training and higher average success rate. To summarize, our method had an average of 40% higher success rate on the "Drone Nav" task, 80% higher success rate on the "Metaworld Reach Wall Task", a 75% higher success rate on the "Metaworld Reach" task, and a 43% higher success rate on the "Farama Push" task.

Lack of Reward Signal. We find that among trajectories of the baseline (SAC without hindsight augmentation), the policy often wanders in a single direction without heading towards the target position / object. This behavior is due to the sparsity of the reward in our environments. Often, the network may never be lucky enough to have encountered a successful trajectory, so the every reward in the replay buffer is just a flat "0" without a reward gradient to improve itself with. Namely in the "Metaworld Reach With Wall" task, never once did the successful trajectory navigate across the wall to the target. Thus, there was no successful trajectories to learn from, leading to a completely unsuccessful policy. Similar phenomena are present in the other environments where there are not enough successful trajectories for the baseline policy to infer effective.

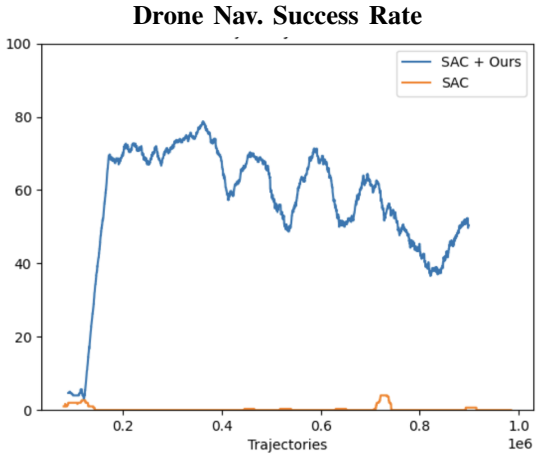

Fig. 7. As seen our method is quite resistant to catastrophic forgetting. Even after running 1 million timesteps, our policy still recovers from the temporary drop in success rate.

Catastrophic Forgetting. Catastrophic forgetting is a phenomenon in reinforcement learning where a policy appears to have failed to achieve a task that it once was capable of achieving. This has been observed in many situations [43]. As seen in the figures, visual hindsight experience replay does not result in significant catastrophic forgetting. We hypothesize that because our method samples subgoals in each trajectory, we maintain a diverse set of goals for our policy to train on. To test how resistant our method is to catastrophic forgetting, we leave our policy running for 1M timesteps and showcase our results in 7. Even though the

success rate dips periodically, the success rate still remains very high and always recovers.

C. Image Editing Speed

As shown in table I, we see that introducing a NeRF into the pipeline does not result in significant slowdown in real world time to train the model. We see that the fps drop from using our hindsight experience replay buffer is almost negligible. Consistent with most visual reinforcement learning works, we use an image resolution ranging from 84 by 84 to 200 by 200. The result of this low image resolution is that our model can render hundreds of frames per second because the render speed is inversely proportional to the number of pixels ie. render_speed $= \mathcal{O}\left(\frac{1}{hw}\right)$. The majority of the time taken in training is actually from the physics engine and training the reinforcement learning network itself. All rendering and calculations were done on a single RTX:A4000 GPU.

TABLE I
FPS WHEN TRAINING (↑)

	Nav. Drone	Reach	Reach W/ Wall	Push
With Hindsight	92	39	46	33
Without Hindsight	89	41	47	39

VI. CONCLUSION AND FUTURE WORK

In this work, we take inspiration from the concept of hindsight reasoning. Learning from hindsight is a crucial step in allowing robots to successful perform tasks and reason about their surroundings. We extend the well-known idea of hindsight experience replay, a method in reinforcement learning that relabels the goals in a trajectory to create synthetic successful trajectories. We show that this method is still effective in the visual domain and can be used in a variety of applications. Future work can extend to other forms of novel view synthesis such as gaussian splatting. Another direction is the use of generative models in place of a novel view synthesis method.

REFERENCES

[1] M. Andrychowicz, D. Crow, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Neural Information Processing Systems*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:3532908

[2] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," 2020. [Online]. Available: https://arxiv.org/abs/2003.08934

[3] P. Wang, Y. Liu, Z. Chen, L. Liu, Z. Liu, T. Komura, C. Theobalt, and W. Wang, "F^2-nerf: Fast neural radiance field training with free camera trajectories," 2023. [Online]. Available: https://arxiv.org/abs/2303.15951

[4] C. Reiser, S. Peng, Y. Liao, and A. Geiger, "Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14 315–14 325, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:232352619

[5] P. Hedman, P. P. Srinivasan, B. Mildenhall, J. T. Barron, and P. E. Debevec, "Baking neural radiance fields for real-time view synthesis," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5855–5864, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:232379923

[6] H. Lin, S. Peng, Z. Xu, Y. Yan, Q. Shuai, H. Bao, and X. Zhou, "Efficient neural radiance fields for interactive free-viewpoint video," *SIGGRAPH Asia 2022 Conference Papers*, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:254044754

[7] J. Gu, A. Trevithick, K.-E. Lin, J. Susskind, C. Theobalt, L. Liu, and R. Ramamoorthi, "Nerfdiff: Single-image view synthesis with nerf-guided distillation from 3d-aware diffusion," 2023. [Online]. Available: https://arxiv.org/abs/2302.10109

[8] N. Pearl, T. Treibitz, and S. Korman, "Nan: Noise-aware nerfs for burst-denoising," 2022. [Online]. Available: https://arxiv.org/abs/2204.04668

[9] Y.-C. Guo, D. Kang, L. Bao, Y. He, and S.-H. Zhang, "Nerfren: Neural radiance fields with reflections," 2022. [Online]. Available: https://arxiv.org/abs/2111.15234

[10] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla, "Nerfies: Deformable neural radiance fields," *ICCV*, 2021.

[11] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, "Plenoctrees for real-time rendering of neural radiance fields," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5732–5741, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:232352425

[12] A. Yu, S. Fridovich-Keil, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance fields without neural networks," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5491–5500, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:245006364

[13] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," 2023. [Online]. Available: https://arxiv.org/abs/2308.04079

[14] J. Luiten, G. Kopanas, B. Leibe, and D. Ramanan, "Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis," in *3DV*, 2024.

[15] 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," 2023. [Online]. Available: https://arxiv.org/abs/2304.02643

[16] M. Ye, M. Danelljan, F. Yu, and L. Ke, "Gaussian grouping: Segment and edit anything in 3d scenes," in *European Conference on Computer Vision*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:265551523

[17] N. Mishra, M. Sieb, P. Abbeel, and X. Chen, "Closing the visual sim-to-real gap with object-composable nerfs," 2024. [Online]. Available: https://arxiv.org/abs/2403.04114

[18] A. Byravan, J. Humplik, L. Hasenclever, A. Brussee, F. Nori, T. Haarnoja, B. Moran, S. Bohez, F. Sadeghi, B. Vujatovic, and N. Heess, "Nerf2real: Sim2real transfer of vision-guided bipedal motion skills using neural radiance fields," 2022. [Online]. Available: https://arxiv.org/abs/2210.04932

[19] M. N. Qureshi, S. Garg, F. Yandun, D. Held, G. Kantor, and A. Silwal, "Splatsim: Zero-shot sim2real transfer of rgb manipulation policies using gaussian splatting," 2024. [Online]. Available: https://arxiv.org/abs/2409.10161

[20] Q. Dai, Y. Zhu, Y. Geng, C. Ruan, J. Zhang, and H. Wang, "Graspnerf: Multiview-based 6-dof grasp detection for transparent and specular objects using generalizable nerf," 2023. [Online]. Available: https://arxiv.org/abs/2210.06575

[21] J. Kerr, L. Fu, H. Huang, Y. Avigal, M. Tancik, J. Ichnowski, A. Kanazawa, and K. Goldberg, "Evo-nerf: Evolving nerf

for sequential robot grasping of transparent objects," in *6th Annual Conference on Robot Learning*, 2022.

[22] E. Zhu, M. Levy, M. Gwilliam, and A. Shrivastava, "Nerf-aug: Data augmentation for robotics with neural radiance fields," 2025. [Online]. Available: https://arxiv.org/abs/2411.02482

[23] A. Zhou, M. J. Kim, L. Wang, P. Florence, and C. Finn, "Nerf in the palm of your hand: Corrective augmentation for robotics via novel-view synthesis," 2023. [Online]. Available: https://arxiv.org/abs/2301.08556

[24] M. Bortolon, T. Tsesmelis, S. James, F. Poiesi, and A. D. Bue, "6dgs: 6d pose estimation from a single image and a 3d gaussian splatting model," 2024. [Online]. Available: https://arxiv.org/abs/2407.15484

[25] ——, "Iffnerf: Initialisation free and fast 6dof pose estimation from a single image and a nerf model," 2024. [Online]. Available: https://arxiv.org/abs/2403.12682

[26] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin, "iNeRF: Inverting neural radiance fields for pose estimation," *arxiv arXiv:2012.05877*, 2020.

[27] A. Rashid, S. Sharma, C. M. Kim, J. Kerr, L. Chen, A. Kanazawa, and K. Goldberg, "Language embedded radiance fields for zero-shot task-oriented grasping," 2023. [Online]. Available: https://arxiv.org/abs/2309.07970

[28] Y. Ze, G. Yan, Y.-H. Wu, A. Macaluso, Y. Ge, J. Ye, N. Hansen, L. E. Li, and X. Wang, "Gnfactor: Multi-task real robot learning with generalizable neural feature fields," 2024. [Online]. Available: https://arxiv.org/abs/2308.16891

[29] N. (Mahi)Shafiullah, C. Paxton, L. Pinto, S. Chintala, and A. Szlam, "Clip-fields: Weakly supervised semantic fields for robotic memory," in *Robotics: Science and Systems XIX*, ser. RSS2023. Robotics: Science and Systems Foundation, July 2023. [Online]. Available: http://dx.doi.org/10.15607/RSS.2023.XIX.074

[30] C.-M. Chung, Y.-C. Tseng, Y.-C. Hsu, X.-Q. Shi, Y.-H. Hua, J.-F. Yeh, W.-C. Chen, Y.-T. Chen, and W. H. Hsu, "Orbeez-slam: A real-time monocular visual slam with orb features and nerf-realized mapping," 2023. [Online]. Available: https://arxiv.org/abs/2209.13274

[31] A. Rosinol, J. J. Leonard, and L. Carlone, "Nerf-slam: Real-time dense monocular slam with neural radiance fields," 2022. [Online]. Available: https://arxiv.org/abs/2210.13641

[32] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2016. [Online]. Available: https://arxiv.org/abs/1511.05952

[33] R. Yang, M. Fang, L. Han, Y. Du, F. Luo, and X. Li, "Mher: Model-based hindsight experience replay," 2021. [Online]. Available: https://arxiv.org/abs/2107.00306

[34] B. Eysenbach, R. Salakhutdinov, and S. Levine, "Search on the replay buffer: Bridging planning and reinforcement learning," 2019. [Online]. Available: https://arxiv.org/abs/1906.05253

[35] C. Lu, P. J. Ball, Y. W. Teh, and J. Parker-Holder, "Synthetic experience replay," 2023. [Online]. Available: https://arxiv.org/abs/2303.06614

[36] R. Wang, K. Frans, P. Abbeel, S. Levine, and A. A. Efros, "Prioritized generative replay," 2024. [Online]. Available: https://arxiv.org/abs/2410.18082

[37] L. Schramm, Y. Deng, E. Granados, and A. Boularias, "Usher: Unbiased sampling for hindsight experience replay," 2022. [Online]. Available: https://arxiv.org/abs/2207.01115

[38] A. C. Li, L. Pinto, and P. Abbeel, "Generalized hindsight for reinforcement learning," 2020. [Online]. Available: https://arxiv.org/abs/2002.11708

[39] T. Yu, D. Quillen, Z. He, R. Julian, A. Narayan, H. Shively, A. Bellathur, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," 2021. [Online]. Available: https://arxiv.org/abs/1910.10897

[40] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. D. Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, H. Tan, and O. G. Younis, "Gymnasium: A standard interface for reinforcement learning environments," 2024. [Online]. Available: https://arxiv.org/abs/2407.17032

[41] B. Ellenberger, "Pybullet gymperium," https://github.com/benelot/pybullet-gym, 2018–2019.

[42] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018. [Online]. Available: https://arxiv.org/abs/1801.01290

[43] M. B. Hafez, T. Immisch, T. Weber, and S. Wermter, "Map-based experience replay: a memory-efficient solution to catastrophic forgetting in reinforcement learning," *Frontiers in Neurorobotics*, vol. 17, June 2023. [Online]. Available: http://dx.doi.org/10.3389/fnbot.2023.1127642